

UNCLASSIFIED

AD NUMBER

AD845169

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies and their contractors; Critical Technology; OCT 1968. Other requests shall be referred to Defense Advanced Research Projects Agency, TIO, Washington, DC 20301. This document contains export-controlled technical data.

AUTHORITY

arpa ltr, 12 may 1971

THIS PAGE IS UNCLASSIFIED

AD845169

MOSAIC - THE IMPROVED EDITING OF SCIENTIFIC  
TEXT BY HAND-DRAWN COMMANDS AND DATA:  
A TECHNIQUE FOR RAND TABLET AND CRT DISPLAY

Seymour R. Friedman  
Douglas A. Campbell  
Leonard G. Fehskens

14 October 1968



COMMAND SYSTEMS DIVISION  
ELECTRONIC SYSTEMS DIVISION  
AIR FORCE SYSTEMS COMMAND  
UNITED STATES AIR FORCE  
L. G. Hanscom Field, Bedford, Massachusetts

Sponsored by: Advanced Research Projects Agency  
Washington, D. C. 20301

ARPA Order No. 766

This document is subject to  
special export controls and each  
transmittal to foreign governments  
or foreign nationals may be made  
only with prior approval of  
Hq ARPA (TIO). *Wash DC*

*20301*



(Prepared under Contract No. AF 19(628)-5992 by Inforonics, Incorporated,  
806 Massachusetts Avenue, Cambridge, Massachusetts 02139.)

**BLANK PAGE**

ADDRESS FOR		
DDSTI	WHITE SECTION	<input type="checkbox"/>
DDO	BUFF SECTION	<input checked="" type="checkbox"/>
UNANNOUNCED		<input type="checkbox"/>
JUSTIFICATION		
BY		
DISTRIBUTION AVAILABILITY CODES		
DIST.	AVAIL. and/or	SPECIAL
2		

### LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

### OTHER NOTICES

Do not return this copy. Retain or destroy.

MOSAIC - THE IMPROVED EDITING OF SCIENTIFIC  
TEXT BY HAND-DRAWN COMMANDS AND DATA:  
A TECHNIQUE FOR RAND TABLET AND CRT DISPLAY

Seymour R. Friedman  
Douglas A. Campbell  
Leonard G. Fehskens

14 October 1968

COMMAND SYSTEMS DIVISION  
ELECTRONIC SYSTEMS DIVISION  
AIR FORCE SYSTEMS COMMAND  
UNITED STATES AIR FORCE  
L. G. Hanscom Field, Bedford, Massachusetts

Sponsored by: Advanced Research Projects Agency  
Washington, D. C. 20301

ARPA Order No. 766

This document is subject to  
special export controls and each  
transmittal to foreign governments  
or foreign nationals may be made  
only with prior approval of  
Hq ARPA (TIO).

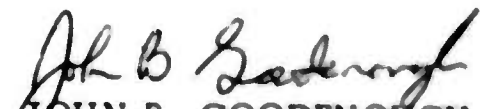
(Prepared under Contract No. AF 19(628)-5992 by Inforonics, Incorporated,  
806 Massachusetts Avenue, Cambridge, Massachusetts 02139.)




## FOREWORD

The work reported here was conducted under ARPA Order 765 by Inforonics, Inc., 806 Massachusetts Avenue, Cambridge, Massachusetts under Contract AF19(628)-5992. The program was monitored by J. B. Goodenough of the Air Force Electronic Systems Division, and was principally performed during the period 14 April 1966 to 14 September 1968. The draft report was submitted on 14 October 1968.

This Technical Report has been reviewed and is approved.

  
JOHN B. GOODENOUGH  
Project Officer

  
WILLIAM F. HEISLER, Colonel, USAF  
Chief, Command Systems Division  
Directorate of Planning & Technology

## ABSTRACT

A method for improved editing of scientific text by hand-drawn commands and data is considered. The most distinguishing feature of the method is that a single quick editing mark over a data segment simultaneously identifies both operator and operand. The method investigated employs a RAND Tablet for the entry of hand-drawn cursive symbols to the computer, an input-only keyboard for large strings of text and a storage-type CRT display unit for displaying the text being edited. The recognition of cursive writing is performed by using geometric-topological invariants of stroke succession. In addition, a design for an editing console is also discussed. In this design, an attempt is made to arrange the different system components to provide for efficient editing. The conclusions derived from the use of the system during its development are presented, along with several areas for further investigation.

**BLANK PAGE**



## TABLE OF CONTENTS

	PAGE
FOREWORD	ii
ABSTRACT	iii
I. INTRODUCTION	1
II. THE MOSAIC SYSTEM	3
2.1 HISTORICAL BACKGROUND	3
2.2 A USER'S VIEW OF THE MOSAIC SYSTEM	3
2.3 THE ANALYSIS PROGRAM	4
III. THE MOSAIC TEXT EDITOR	5
3.1 INPUT PROCESSING	5
3.2 BUFFER AREAS	5
3.3 OUTPUT PROCESSING	8
3.4 THE CRT STORAGE DISPLAY	8
3.5 THE INPUT KEYBOARD	8
3.5.1 CONTROL MODE PROCESSING	8
3.5.2 TEXT MODE PROCESSING	10
3.6 THE RAND TABLET	10
3.7 KEYBOARD/TABLET INTERACTION	10
3.8 THE TELETYPEWRITER	10
IV. THE RECOGNITION ALGORITHM	12
4.1 CURSIVE WRITING	12
4.2 THE CHARACTER OF STROKES	13
4.3 OVERVIEW OF THE RECOGNITION PROCESS	14
4.4 INPUT OF POINTS FROM THE TABLET	14
4.5 STROKE CLASSIFICATION METHOD	15
4.5.1 STROKE SEGMENTATION	16

## TABLE OF CONTENTS (Continued)

	PAGE
4.5.2 SHAPE CODING	17
4.5.2.1 STROKE LENGTH	17
4.5.2.2 STROKE DIRECTION	17
4.5.2.3 STROKE CURVATURE	18
4.5.3 RELATION CODING	20
4.5.3.1 STROKE RETURN AND CLOSURE (y-y OR BREAK-y PAIRS ONLY)	20
4.5.3.2 STROKE INCIDENCE AND CROSSING	21
4.6 ARCHETYPE CODING	22
4.6.1 PHILOSOPHY	22
4.6.2 SHAPE CODE MATCHES	22
4.6.3 RETURN AND CLOSURE CODE MATCHES	22
4.6.4 INCIDENCE AND CROSSING CODE MATCHES	23
4.7 SOME PROBLEMS WHICH DEVELOPED WITH THE RECOGNITION METHOD AND THEIR SOLUTIONS	23
4.7.1 NOISE AND SMOOTHING	23
4.8 AREAS FOR FUTURE INVESTIGATION	25
4.8.1 AN ALTERNATE AVERAGING SCHEME	25
4.8.2 HORIZONTAL SEGMENTING	25
4.8.3 QUADRANT SEGMENTING	26
4.8.4 REORDERING OF THE DICTIONARY	27
4.8.4.1 PROBABILISTIC ORDERING	27
4.8.4.2 LIST STRUCTURING	28
4.8.5 AN ALTERNATE RECOGNITION METHOD: RECOGNITION BY MAJOR SUB-FEATURES	28
V. THE EDITING CONSOLE	31

# TABLE OF CONTENTS (Continued)

	PAGE
VI. CONCLUSIONS	32
6.1 THE RAND TABLET	32
6.2 SYMBOLIC CONTROL: A BASIC TABLET TECHNIQUE	32
6.3 THE STORAGE-TYPE CRT DISPLAY UNIT	33
6.4 THE EDITING CONSOLE	33
APPENDIX I. MOSAIC-OPERATION AND COMMANDS	35
APPENDIX II. FUNCTIONAL FLOW CHART OF TEXT EDITOR	41
APPENDIX III. LAYOUT OF THE INPUT KEYBOARD	44
APPENDIX IV. THE TELETYPE MESSAGES	47
APPENDIX V. FUNCTIONAL FLOW CHART OF RECOGNITION ALGORITHM	48
APPENDIX VI. STROKE CODES	49
APPENDIX VII. DICTIONARY ARCHETYPE CODES	53
APPENDIX VIII. FLOW CHART OF HORIZONTAL SEGMENTING ALGORITHM	62
BIBLIOGRAPHY	64

## LIST OF ILLUSTRATIONS

	PAGE
FIGURE 1. INPUT RECORD OF PSYCHOLOGICAL ABSTRACTS - PRODUCTION DATA USED FOR TESTING MOSAIC	2
FIGURE 2. SAMPLE INPUT TEXT CONTAINING "AUTO-EDITING" COMMANDS	6
FIGURE 2A. THE SAMPLE INPUT TEXT AFTER "AUTO-EDITING"	7
FIGURE 3. THE CRT STORAGE DISPLAY	9
FIGURE 4. CUSP AND CORNER	12
FIGURE 5. INCIDENCE AND CROSSING	14
FIGURE 6. DICTIONARY STROKE CODE WORD	16
FIGURE 7. STROKE SEGMENTATION	16
FIGURE 8. STROKE DIRECTION	17
FIGURE 9. STROKE COORDINATE ASSIGNMENTS	19
FIGURE 10. STROKE CONDITIONS FOR CLOSURE, RETURN AND OPEN RETURN	20
FIGURE 11. STROKE CLASSIFICATIONS BY DIRECTION	26
FIGURE 12. THE EDITING CONSOLE (ARRANGEMENT FOR A RIGHT-HANDED USER)	31
FIGURE 13. THE TABLET CONTROL AREAS	37
TABLE 1. SHAPE CODES	49
TABLE 2. RETURN AND CLOSURE	51
TABLE 3. INCIDENCE AND CROSSING CODES	52

## INTRODUCTION

During the past few years, we have been developing a new method of editing machine-form text, in which proofreaders' marks and changes are made directly on the data. These marks and changes are interpreted by pattern recognition programs and are converted to machine commands which are then executed. This technique uses a RAND Tablet for hand-drawn input and a storage-type cathode ray tube display unit (CRT) for text display.

The system name, MOSAIC, is a descriptor rather than an acronym and is intended to convey the important two-dimensional aspect of system operation. The most distinguishing feature of MOSAIC is that a single quick editing mark over a data segment simultaneously identifies both operator and operand. It is the recognition of these control-location symbols that provides the major power of the system. The recognition of handwritten alpha-numeric items, while necessary for a uniform penscript driven editing system, is not in itself an efficient substitute for type-written entry of text. Thus, in MOSAIC, large blocks of text may be entered via an input keyboard.

The MOSAIC technique has yielded results in two main areas:

- (a) Methods for more efficient text editing
- (b) Determination of the principle advantages of the RAND Tablet in data manipulation

These results will be discussed in detail on the following pages.

The hardware configuration used for the MOSAIC System consists of the following components:

- (a) PDP-9 with 8k of core memory, high-speed paper tape reader and punch
- (b) Grafacon RAND Tablet, Model 1010
- (c) Tektronix Storage Display Unit, Model #11
- (d) Invac Keyboard, Model PK-144
- (e) Teletypewriter, Model 33 KSR

020	9.8.2						
Ajmal, Muhammad							
Acceptance in psychoanalysis.							
Journal of Psychology Lahore, 1966, MR3R(1), 1 10.							
Describes a method to accept and understand the feelings of the client: (1) induction of a process of acceptance in the analyst by imaginative reversal, and (2) deliberate and willful formation of images by the analyst depicting the condition of the client. A 17620, client feelings, acceptance & understanding 17440, client feelings, acceptance & understanding							

00027	9.2.2						
Cassini, Eugenio							
L'Arte come espressione clinica. Art as clinical expression.							
Rivista di Psicoanalisi 1965, ANIIM(3), 227 233.							
Artistic expression is related to various aggressive and sexual instincts. L. Libate.							
10790, clinical expression, aggressive instincts, sexual instincts							
10410, art, clinical expression, aggressive instincts, sexual instincts							
14800, art, clinical expression, aggressive instincts, sexual instincts							
18780, art, clinical expression, aggressive instincts, sexual instincts							

Figure 1. Input Record of Psychological Abstracts -  
Production Data Used for Testing MOSAIC

## II

### THE MOSAIC SYSTEM

#### 2.1 HISTORICAL BACKGROUND

The work on an on-line editing technique using hand-drawn commands and data was carried out in two phases. In the initial phase, the original technique was developed and tested. The results of the tests performed provided the necessary direction in developing an improved technique, which is the subject of this report and forms the second phase of the work. The contents of this report include the following:

- (a) A general description of the MOSAIC system
- (b) The text editor
- (c) The character recognition algorithm
- (d) The editing console

Our systems philosophy has been to emphasize directness. When an editor or proofreader marks up page copy, he makes marks directly on the data and does not concern himself with location concepts, e.g., "2nd paragraph, 3rd line, 7th word;" he simply places his blue pencil on the spot where correction is to be made and makes an appropriate mark there. Neither does he concern himself with concepts of data extent or data names, e.g., "lines, word, letter;" he simply makes a mark of appropriate type or extent. This kind of directness is quite simple, but is totally lacking in other machine-editing systems. Some early light-pen systems used a "pointing" technique, but the command was never combined with selection. In the MOSAIC system, all commands and data can be entered by writing on a RAND Tablet.

#### 2.2 A USER'S VIEW OF THE MOSAIC SYSTEM

Original keying of the manuscript copy of scientific journals is performed by a keyboarder on a Dura Keyboard in batches of 25 records. For our testing we used actual production samples of data prepared for publication in Psychological Abstracts. The Dura page copy of the input record is shown in Figure 1. This typed copy is proofread by comparison with the manuscript copy, and corrections are noted on the typescript page in ink. This typescript now accompanies the Dura tape to the computer, the tape is inserted in the reader, and on-line editing is ready to begin.

The operator initially enters commands via the input keyboard. When the tape to be edited is read in, the first portion of it is displayed on the CRT.



To enter the mode for tablet use, the operator types the "PE" command. This activates the tablet pen. MOSAIC starts in Control Mode in which marks are interpreted as commands. When the command is one that requires text entry, the program shifts to Text Mode, in which marks are interpreted as alphanumeric symbols and punctuation. In the latter mode, the message "TEXT MODE" is displayed at the top of the CRT. In this mode, text may be entered from either the tablet or the keyboard, where the latter device would be used in the case of a large amount of text.

The editor reads marked-up typescript, and edits and advances through the machine-form text. The CRT display provides immediate verification of correct editing action. If a hand-drawn symbol cannot be interpreted, a message is typed on the computer teletype, and the editor redraws the symbol. The symbol dictionary provides definitions of each symbol in terms of the size-invariant stroke composition method used for recognition, which is discussed in Section IV. This recognition method has proven effective and fast, though obviously dependent on the allowable dictionary size. Non-recognition, however, does not cause error introduction, but only longer editing time due to repetition. The operator always concentrates on the CRT display, and does not watch his hand motion on the tablet, so that there is no lost time in attention switching from the writing medium to the display medium. Manual paper handling of the typescript is still required, however, to find the location of the next needed correction. In addition, where large blocks of text are to be entered, the editor may use the input keyboard.

When all the text which was read in is completely edited, it is punched out on paper tape and more text is read in. This process is continued until all of the input text is edited.

### 2.3 THE ANALYSIS PROGRAM

The development of symbol recognition methods has been aided substantially by the Analysis program. This program accepts hand-drawn symbols as input, displays them, performs feature extraction upon them, matches them against the current version of the MOSAIC symbol dictionary, and types out the recognition decision. Whether or not a symbol is recognized, the program causes typeout of the derived feature list of the symbol in simple English form. By entering a large body of samples into the system, the prominent and/or invariant features of a new symbol can be quickly perceived, and new dictionary entries can be quickly generated.



### III

#### THE MOSAIC TEXT EDITOR

The text editing capabilities of the MOSAIC System include the insertion, deletion, rearrangement (transposition), and changing of strings of characters. A complete list of available commands appears in Appendix I, and a functional flow chart in Appendix II. In this section, the various components of the MOSAIC editing program will be discussed.

#### 3.1 INPUT PROCESSING

A request for more paper tape input can be initiated either from the tablet or from the keyboard. Unique commands allow the user to control the amount of text to be input, (i.e., up to a previously designated character sequence) and to decide whether to delete the previous text or to append to it.

Input characters are checked for legality and proper parity. Erroneous characters produce an error message on the KSR teleprinter and are replaced in the text buffer by a special character which shows up clearly on the CRT.

"Auto-editing" is performed to delete errors noted by the keyboard, when the text was originally prepared. The following is the coding scheme employed in "auto-editing."

- (a)  $\langle \text{backspace} \rangle_1 \langle \text{backspace} \rangle_2 \dots \langle \text{backspace} \rangle_n \langle \diagup \rangle_1 \langle \diagup \rangle_2 \dots \langle \diagup \rangle_n$  will delete the n previous characters from the input.

- (b)  $\langle \square k_l \rangle$  will delete the previous line.

- (c)  $\langle \square k_r \rangle$  will delete the previous record, (i.e., all the preceding text to the sequence of character pre-designated by the user as the terminator for a logical record). See Figures 2 and 2A.

#### 3.2 BUFFER AREAS

There are three buffers used by the MOSAIC editing program:

- (a) The Text Buffer - In this buffer, all input, output, and editing operations (inserting, deleting, changing characters) are performed.
- (b) Record Delimiter Buffer - In this buffer, the user may store the string of characters which denotes the end of a logical record.

01 10751  
 12 Malmstrom~~kl~~  
 02 Malmstrom  
 03 Dr  
 04 Edward J. ~~XXXXXXXXXX~~(James)  
 06 June 5, 1932  
 07a BA 59  
 08a UC Berkeley  
~~kl~~  
 08a UC Berkeley  
 07b MA 63  
 07c PhD 66  
 08c U Portlan ~~XXXXXXXXXX~~  
 09 Resch asst 59-60, resch asst Survey  
 Resch Cen 62-63 UC Berkeley; trnee 61-62  
 VAM, San Francisco; resch asst Cardiovascular  
 Resch Inst 62-63 UC San Francisco; psych exam~~kl~~  
 63-65 Portland Pub Schs, Ore; VRA post doc  
 resch fel 66-67  
 10a post/doc resch fel Dept Psychiat & Brain Resch  
~~XXXXXX/XXXXXX/XXXXXX~~Inst 67|  
 11a UCLA  
 10b resch psychophysiol 67|  
 11b Cedars-Sinai Hosp  
 12a Psychophysiol  
 12b stat methodology  
 12c persy theory  
 13 444 Landfair Ave  
 14 Los Angeles  
 15 Calif  
 16 90024  
 24 68  
 30 68  
~~kl~~  
 01 10752  
 02 Booth  
 04 Linda Ellen  
 06 See Rapport, Linda Ellen  
~~kl~~  
 01 10753  
 02 Benjamin  
~~kr~~  
 01 10754  
 02 Bennett  
 03 Dr  
 04 Barbara (Ann)  
 06 See Thomas, Dr Barbara R.  
~~kl~~

Figure 2. Sample Input Text Containing "Auto-Editing" Commands

01	10751
02	Malmstrom
03	Dr
04	Edward J James)
06	June 5, 1932
07a	BA 59
08a	UC Berkeley
07b	MA 63
07c	PhD 66
08c	U Portlan d
09	Resch asst 59-60, resch asst Survey
	Resch Cen 62-63 UC Berkeley; trnee 61-62
	VAH, San Francisco; resch asst Cardiovascular
	63-65 Portland Pub Schs, Ore; VRA post doc
	resch fel 66-67
10a	post/doc resch fel Dept Psychiat & Brain Resch.
Inst 67	
11a	UCLA
10b	resch psychophysiol 67
11b	Cedars-Sinai Hosp
12a	Psychophysiol
12b	stat methodology
12c	persy theory
13	444 Landfair Ave
14	Los Angeles
15	Calif
16	90024
24	68
30	68
■	
01	10752
02	Booth
04	Linda Ellen
06	See Rapport, Linda Ellen
■	
01	10754
02	Bennett
03	Dr
04	Barbara (A.M.)
06	See Thomas, Dr Barbara B.
■	

Figure 2A. The Sample Input  
Text After "Auto-Editing"

- (c) Search Buffer - In this buffer, the user may store a string of characters which can later be used by the various search routines to locate occurrences of this same string in the Text Buffer.

All three buffers are always displayed on the CRT.

### 3.3 OUTPUT PROCESSING

A request to punch out the present contents of the Text Buffer can be initiated from either the tablet or the keyboard. Separate commands allow various formats, such as text only, text with a stop code, text with a section of blank tape, etc. The output routines also eliminate any extraneous characters, such as redundant case shifts.

### 3.4 THE CRT STORAGE DISPLAY

Figure 3 illustrates the CRT storage display used in the MOSAIC System. Since the CRT used is of the storage type, a new display is initiated only when the user enters a new command from either the keyboard or tablet. A 5 x 7 dot matrix is used to display a character on the CRT.

All lines displayed on the CRT begin with a line number generated by MOSAIC at the far left, followed by up to 64 text characters. When the 64 character limit is exceeded, a new line (and line number) is automatically generated. Up to 24 lines of the Text Buffer, followed by the Record Delimiter Buffer and lastly the Search Buffer (at the bottom of the CRT), are displayed with the line numbers starting at 01 for each buffer.

### 3.5 THE INPUT KEYBOARD

An input-only keyboard is used by the MOSAIC System to enter both control commands and (large) text strings. Although the keyboard is a single-case keyboard, the software is able to simulate the upper/lower case Dura keyboard, on which the original data was created. (See Appendix III.) The manner in which input from the keyboard is handled is dependent upon the mode in which the system is operating, i.e., whether it is in Control Mode or Text Mode.

#### 3.5.1 CONTROL MODE PROCESSING

In the Control Mode, alphabetic characters which are received from the keyboard are used to determine the editing operation to be performed. Numeric characters are accumulated and provide the argument for the operation. Upon receiving a "return" character, the execution of the operation is initiated.



Figure 3. The CRT Storage Display

### 3.5.2 TEXT MODE PROCESSING

In the Text Mode, all characters received from the keyboard are interpreted as text, and are stored in the buffer required by the current command. Return to Control Mode results upon the entry of two sequential case shifts.

### 3.6 THE HAND TABLET

The user must enable the tablet by entering a "PE" command at the keyboard. There is a 1:1 correspondence between the CRT display area and the tablet writing area. When the pen traces lightly over the tablet, a light trace of the pen position is superimposed over the text display. When the user presses down on the pen, the pen-tip micro-switch is closed and the marks are interpreted as editing commands or text, until the pressure on the pen is released. Only 2.5 ounces of force are required to close the switch, so no unnatural writing pressure is required. The editing commands are drawn directly over the displayed text upon which the operations are to be performed.

The MOSAIC editing program first checks if the symbol (regardless of what it is) was drawn on one of the eight control areas on the tablet, i.e., the four corners and the four middle edges. If so, no further character recognition is necessary, and the corresponding operation is performed. The layout of the tablet and the operations corresponding to the control areas are described in Appendix 1. If the character was not drawn on one of the control areas, the character recognition routines return with a Dura code, if the input point sequence matches a character in the recognition dictionary. Otherwise, a message is printed on the teleprinter.

### 3.7 KEYBOARD/TABLET INTERACTION

MOSAIC is designed so that the command processing is the same whether characters were entered from the keyboard or from the tablet. The user could initiate, for example, an "insert characters" operation at the tablet, draw a few characters, switch to the keyboard to type in more characters, and return to the tablet to execute the end-of-text command.

All basic paper tape input/output and display control commands can be initiated at either the keyboard or the tablet. Editing commands which must "point" to a character or string of characters can be initiated only from the tablet.

### 3.8 THE TELETYPEWRITER

The teletype is used strictly as an output device to type short messages, in most cases informing the user of an error

condition. Any teletype input is ignored by the system. The teletype messages printed by MOSAIC appear in Appendix IV.



## IV

### THE RECOGNITION ALGORITHM

The recognition method<sup>1</sup> makes maximal use of the information given by the sequence of strokes which constitute the cursive symbols drawn. The strokes are classified according to their geometric properties and their topological relation to predecessor strokes. When archetype or standard symbols are defined in terms of the necessary geometric and topological properties of constituent stroke sequences, the classification of input is invariant under gross distortion. As the method is designed for real-time recognition, the number of extracted properties is minimized.

#### 4.1 CURSIVE WRITING

Cursive writing is fundamentally a system for continuous writing using flowing lines of more or less continuous curvature. It has the singular characteristic of having no nodes (deflection points of multiple tangency). All direction changes occur via continuous curves or via cusps (deflection points of single tangency). Hence corners, in the usual sense, do not occur in cursive writing, whereas, they are an important property of block letters. (See Figure 4.)



Figure 4. Cusp and Corner

---

1. W.R. Nugent, The On-Line Recognition of Cursive Writing Using Geometric-Topological Invariants of Stroke Succession, First Annual IEEE Computer Conference, Chicago, Illinois, September 1967.



## 4.2 THE CHARACTER OF STROKES

There are philosophical and practical reasons for dividing cursive symbols into strokes. Philosophically, the most significant elements of writing seem to consist of excursions, or strokes, roughly along a North-Northeast axis, but only rarely do they contain inflection points. When inflection points exist, they tend to represent insignificant attributes of capital letters, which are not recognized in this system, or the singular case of the numeral 8. Excursions that are primarily in the x direction, or x strokes, are also rare and are relatively insignificant in lower case letters. When x strokes occur in numerals, they are usually straight.

There are also practical reasons for dividing input symbols into strokes. The general cursive symbol is a multiple valued function. A stroke, as we have defined it, is single valued in its major axis, and hence easier to process.

The essential theory of the recognition method constitutes a theory of symbol construction in cursive writing; some strokes primarily must have specified shapes, while others primarily must have specified geometric or topological relations to their predecessors.

Strokes may be conveniently classed in three sizes: major, minor, and micro. A major stroke extends to a length approximately the height of the symbol; a minor stroke is discernibly smaller, but still a significant constituent; and a micro stroke is extremely small and often, but not always, part of an insignificant flourish or hook.

A most important property of the y stroke is that of their pairwise dependence. There is a high probability that the second stroke of a successive pair will have one or more geometric or topological relations to its predecessor. While not a surprising property, it is a most useful one for recognition. The basic geometric relation is that of return to the y level. This occurs when the end point of a stroke has approximately the same y coordinate as the origin of the previous stroke. This condition can be further characterized by the distance  $d$  between two points; closure exists when  $d \leq \beta$ , an open return exists when  $d \geq \alpha$  specified distance, and an indeterminate condition of return-only exists when  $d$  is between these limits. All distances are relative, and are expressed as a fraction of the height of the enclosing rectangle.

The most important topological relations with respect to the previous stroke are those of incidence and crossing. Incidence in our usage consists of any extremely close proximity. (See Figure 5.) Incidence and crossing are further classified as to which third in y of the previous stroke the crossing occurred.



Figure 5. Incidence and Crossing

When the strokes of an input symbol are characterized by their geometric-topological properties, a condensed and sufficient representation results that may be conveniently coded.

#### 4.3 OVERVIEW OF THE RECOGNITION PROCESS

Tablet input is enabled by a keyboard command from the user. Points are processed by an input routine and stored. When the point sequence is judged complete, input is terminated, and the editing program portion of the system calls on the recognition programs. These programs examine the point sequence and break it up into strokes, which are further characterized as to their properties. The sequence of strokes is then compared with a dictionary of defining archetypes, and if the character is found, the appropriate character code is returned to the editing program. If the character is not found, a special "not recognized" code is returned. A functional flow chart of the recognition algorithm is provided in Appendix V.

#### 4.4 INPUT OF POINTS FROM THE TABLET

The x and y coordinates of the pen position on the tablet are read 60 times a second. This is handled by a real time clock, set to request a program interrupt at that rate. The coordinates are compared with the last set of coordinates accepted; if the point is not more than four coordinate units (0.04") distant in either x or y from the previous point, the point is rejected, and the interrupted program is restored. If the point is more than 20 coordinate units (0.2") distant in either x or y, it is rejected as noise. Distance parameters can be changed if necessary.

If the point is accepted, it is used to update a four point moving average, and the averaged value is stored in a table.

This process is repeated until the table is filled or the pen is lifted from the tablet. The latter condition is indicated by a special bit in one of the tablet coordinate registers. When the table is full, the clock is turned off, and a flag indicating a completed symbol input is set. When the pen is lifted, the input program monitors the tablet for a number of clock breaks, and if the pen does not return to the tablet, the symbol is considered completed. If within the specified time, the pen begins tracing a new sequence on the tablet, the process of examining points is begun again. A special marker is set in the coordinate table, indicating where in the point sequence the pen was lifted. This time period allows the pen to be lifted to draw symbols with disconnected strokes, as for example, the dot of an "i" and the bar of a "t". Once a symbol is considered completed, the clock is turned off, and a flag indicating symbol completion is set. The interrupted program is then restored.

#### 4.5 STROKE CLASSIFICATION METHOD

The stroke classification method employed operates on the point sequence representing a symbol. For our purpose, the smoothed output of the clock-sampled tablet coordinates is used. The following operations are performed:

- (a) Stroke segmentation
- (b) Classification of stroke according to length
- (c) Classification of stroke according to major axis of traversal
- (d) Classification of stroke according to direction
- (e) Classification of stroke according to curvature
- (f) Classification of stroke according to return and closure properties, with respect to the previous stroke
- (g) Classification of stroke according to incidence and crossover properties, with respect to the previous stroke

A sequence of stroke codes defines a specific input symbol. The defining archetype symbol in the dictionary is similarly coded, but in a more general way, such that an archetype property may be defined as a set of disjunctive input properties.

These classifications are coded into a single computer word per stroke. Bits within a stroke code word have the meanings depicted in Figure 6.

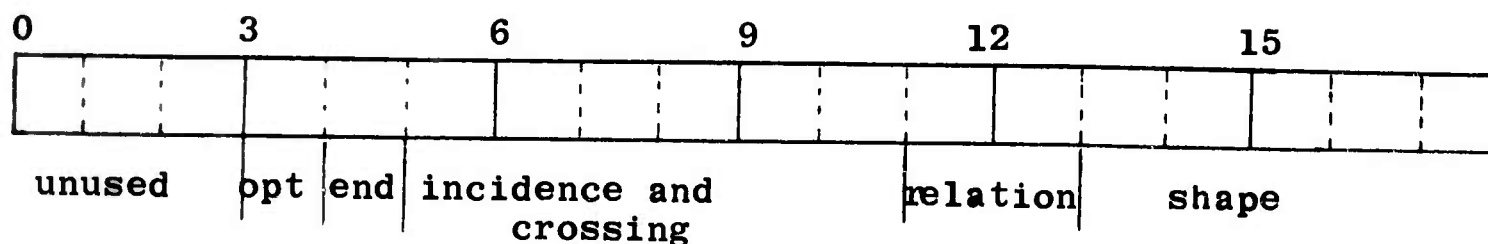


Figure 6. Dictionary Stroke Code Word

The opt code is used only in dictionary archetypes. If it is set, the stroke represented by this word is optional and need not occur in the input stroke sequence. If the end bit is set, the word represents the last stroke of the stroke sequence. In the dictionary, the low order six bits (12-17) of this word contain a code representing the character. In an input code word, these six bits are unused.

#### 4.5.1 STROKE SEGMENTATION

Strokes start at the beginning of a symbol, at the beginning of disjoint elements, and whenever there is a change in direction between two successive runs. A run is defined as a sequence of four or more points moving in the same (vertical) direction. In this case, the first point of the second run is considered the break point, and becomes the last point of the previous stroke and the first point of the present stroke (See Figure 7.)

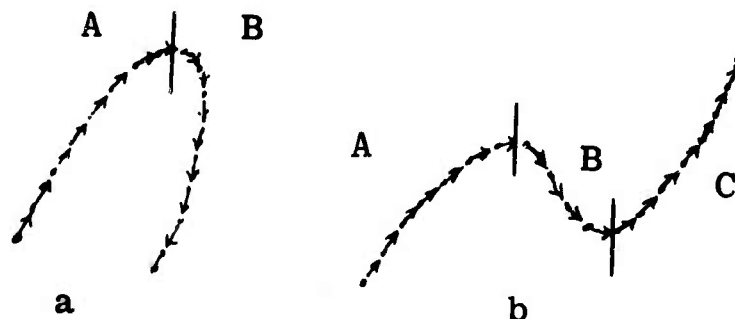


Figure 7. Stroke Segmentation

In Figure 7a, both sequences A and B qualify as runs, and thus a breakpoint is defined at their common point. In Figure 7b, although sequences A and C qualify as runs, sequence B does not. Thus, no breakpoints are defined and no segmentation takes place. Note that if B were a run, breakpoints would be defined at both the common point of A and B and the common point of B and C.

#### 4.5.2 SHAPE CODING

Information about size, direction, and curvature of a stroke are subsumed by a five bit shape code. These codes are listed in Table 1 of Appendix VI.

##### 4.5.2.1 STROKE LENGTH

The distance,  $d$ , between the end points of the stroke,  $S$ , is determined and compared with  $Y$ , the total  $y$  dimension of the symbol.

If  $d \geq 70\%Y$ , then  $S$  is a Major Stroke

Otherwise, if  $d \geq 15\%Y$ , then  $S$  is a Minor Stroke

Otherwise,  $S$  is a Micro Stroke with a shape code "25"

##### 4.5.2.2 STROKE DIRECTION

Non-Micro Strokes are categorized as  $x$  strokes or  $y$  strokes depending on their slope, with a bias toward  $y$  strokes. Any stroke with a slope of  $35^\circ$  or greater is considered a  $y$  stroke, that is, considering the end points of the stroke and their  $x$  and  $y$  distances  $\Delta x$  and  $\Delta y$ :

If  $\left| \frac{\Delta y}{\Delta x} \right| \geq .6$ , then  $S$  is a  $y$  stroke

Otherwise,  $S$  is an  $x$  stroke

$x$  strokes and  $y$  strokes are classified as  $+$  or  $-$ , depending on the drawing sequence of their end points. (See Figure 8.)

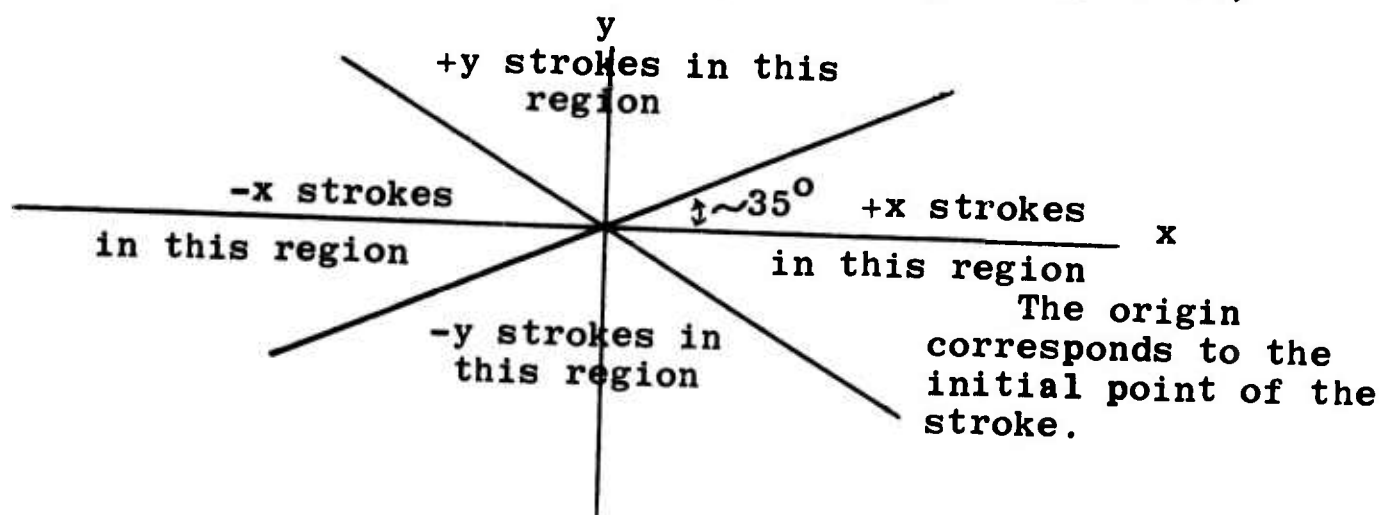


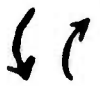
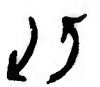
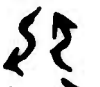
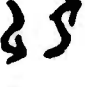
Figure 8. Stroke Direction



#### 4.5.2.3 STROKE CURVATURE

Curvature is calculated for y strokes only. A "+1" curvature is assigned to strokes which extend to the right of the line connecting their end points, a "-1" curvature is assigned to strokes which extend to the left of the line, and a "0" is assigned to strokes with no appreciable extension. If an inflection point exists, a "+2" or "-2" curvature is assigned, depending on whether the initial curvature is "+" or "-". Only major strokes are coded for inflected curvature; if the condition is detected for a minor stroke, the initial curvature is assigned.

Examples of curvature are as follows:

-  -1; stroke extends to left of chord
-  +1; stroke extends to right of chord
-  -2; initially, stroke extends to left of chord
-  +2; initially, stroke extends to right of chord

Curvature is computed by calculating distances in x from the chord linking the end points of the stroke to the actual arc of the stroke. Three sample points, equidistant in y, are taken on the chord. The magnitude and direction of these distances are used to determine curvature. (See Figure 9.)

- (a) The initial point of the stroke  $p_1 = (x_1, y_1)$  and the point  $p_2 = (x_2, y_2)$  are taken and the stroke extent in y,  $\Delta y_s$ , is determined:  $\Delta y_s = y_1 - y_2$
- (b) Determine the three interior quarters of the extent in y to be used as chord coordinates:  $y_{c1}, y_{c2}, y_{c3}$ .

$$y_{c1} = y_2 + \frac{3\Delta y_s}{4}$$

$$y_{c2} = y_2 + \frac{2\Delta y_s}{4}$$

$$y_{c3} = y_2 + \frac{\Delta y_s}{4}$$

- (c) Find the slope, m, of the stroke chord.

$$m = \frac{y_1 - y_2}{x_1 - x_2}$$

- (d) Find the three x coordinates,  $x_{c1}$ ,  $x_{c2}$ ,  $x_{c3}$ , of chord points corresponding to  $y_{c1}$ ,  $y_{c2}$ ,  $y_{c3}$ , where

$$x_{ci} = \left( \frac{y_{ci} - y_1}{\Delta y_s} \right) \cdot x_1; \quad i = 1, 2, 3$$

- (e) Find the three x coordinates,  $x_{a1}$ ,  $x_{a2}$ ,  $x_{a3}$ , of arc points corresponding to  $y_{c1}$ ,  $y_{c2}$ ,  $y_{c3}$ . Symbol points are searched to find the closest points to  $y_{c1}$ , and the x coordinates of these points are used.

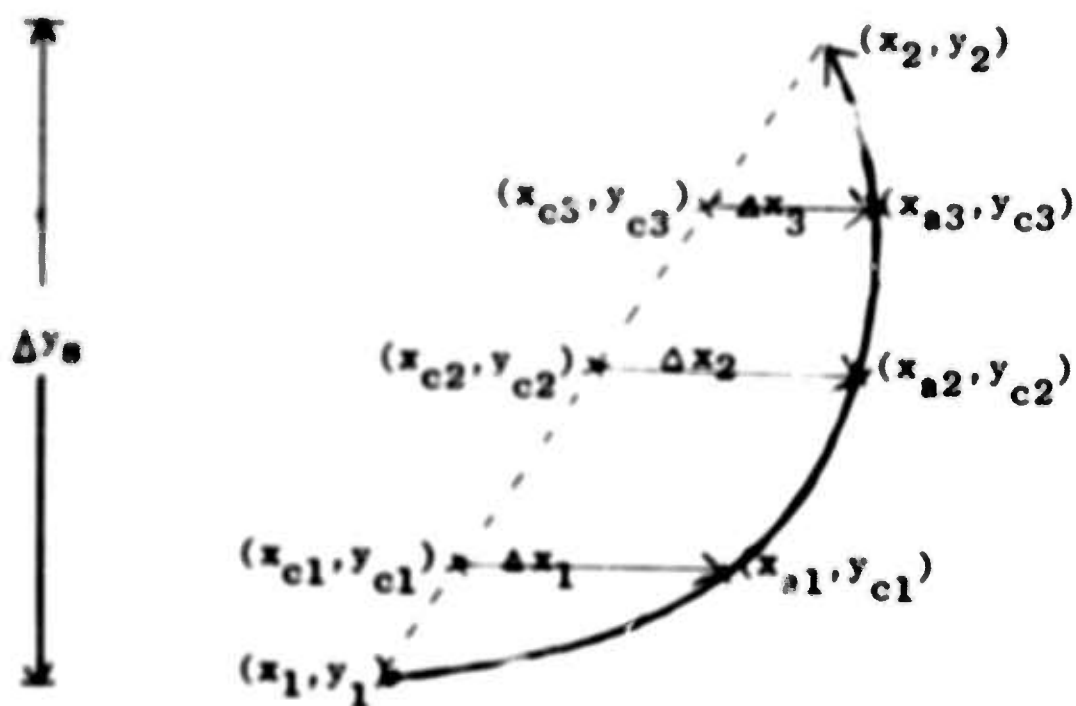


Figure 9. Stroke Coordinate Assignments

- (f) Find  $\Delta x_i = x_{ai} - x_{ci}$ ;  $i = 1, 2, 3$
- (g) If the stroke moves downward, the values of  $\Delta x_1$  and  $\Delta x_3$  are interchanged, so that  $\Delta x_1$  corresponds to the difference at the bottom quarter,  $\Delta x_2$  to the middle, and  $\Delta x_3$  to the top quarter chord points.
- (h) Eliminate any  $\Delta x_i$  if  $|\Delta x_i| \leq 10\% \Delta y_s$
- (i) Of those  $\Delta x_i$  remaining, check signs, and if all signs are alike, then apply the curvature label "+1" if plus, or "-1" if minus. If signs are different, apply the curvature label "+2" if the first  $\Delta x$  is plus, "-2" if

the first  $\Delta x$  is minus, considered in the order  $\Delta x_1, \Delta x_2, \Delta x_3$ .

#### 4.5.3 RELATION CODING

The geometric and topological relations to be detected and coded deal with stroke return, closure, incidence, and crossing properties of strokes, and occupy an eight bit code.

##### 4.5.3.1 STROKE RETURN AND CLOSURE (y-y OR BREAK-y PAIRS ONLY)

Stroke return occurs when the final point  $P_f = (x_f, y_f)$  of the present stroke has a y coordinate close to that of the starting point  $P_i = (x_i, y_i)$  of any previous stroke. Closure occurs when, in addition to return, the x coordinates of these two points are also proximate.

Calling  $Y_p$  the y magnitude of a previous stroke, a simple return condition,  $r$ , exists under the following conditions:

$r$  if  $|y_f - y_i| \leq 10\% Y_p$ ; otherwise  $\emptyset$

A closure condition,  $C$ , exists as follows:

$C$  if  $r$  and  $|x_f - x_i| \leq 10\% Y_p$

If return but not closure exists, an open return,  $U$ , may exist as follows:

$U$  if  $r$  and  $|x_f - x_i| \geq 20\% Y_p$ , otherwise

$R$  if  $r$  and  $\bar{C} \bar{U}$ .

Figure 10 depicts those areas in which the final point of the present stroke must fall for a given condition to hold.

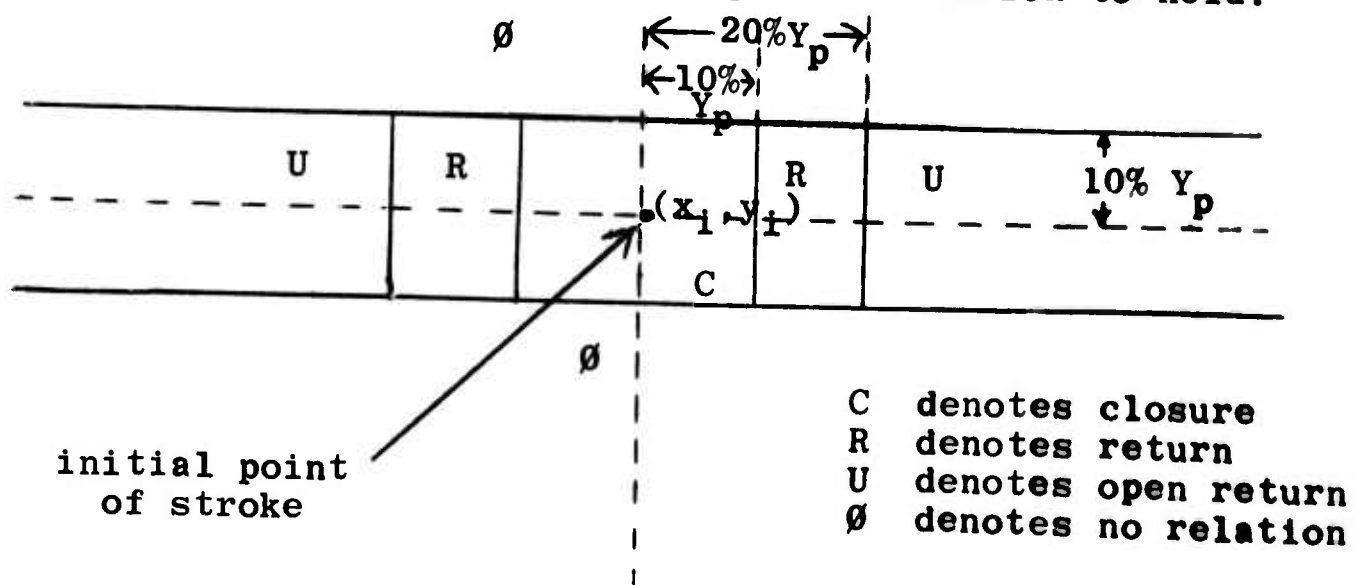


Figure 10. Stroke Conditions for Closure, Return and Open Return



The coordinates of the initial point of each stroke are stored in a table, as well as 10% of the magnitude of the y extent of the stroke, each time a breakpoint is defined. The earliest strokes are stored first, and the most recent are stored last in the table.

When determining the return condition for a stroke, the table is examined backwards; that is, the most recent strokes are examined first. Closure is given priority, in that, as soon as a closure condition is found, the search stops, and a closure condition is assigned to the stroke.

If closure is not found with any previous stroke, the table is examined again backwards, and the first return condition of any sort found is assigned to the stroke. If no return is found, the "none" condition is assigned.

The return and closure conditions are indicated by a two bit code. These codes are listed in Table 2 of Appendix VI.

#### 4.5.3.2 STROKE INCIDENCE AND CROSSING

The incidence and crossing relations to be detected indicate whether the present stroke impinges on or crosses the previous stroke. The y length,  $Y_p$ , of the previous stroke is divided into thirds, and the point of incidence or crossing is noted. Seven conditions are detected, as follows:

0: No incidence or crossing

$X_1, X_2, X_3$ : Crossing at top, mid, or bottom third, whichever is detected first.

$I_1, I_2, I_3$ : No crossing detected in any third, but incidence at top, mid, or bottom third, whichever is detected first.

The points of the two strokes are compared in an order following the up or down direction of the present y stroke. The closest y coordinates are compared, and only the common y range of the two strokes is used. Checks are made to be sure that the strokes do indeed share a common range. If the two strokes share a common point, as in the most frequent case, the range from this point to a point 15% of  $Y_p$  is ignored. Within the remaining range, each point of the previous stroke is paired with the point closest in y on the present stroke. The difference in x coordinates is taken. The difference must at one point have an absolute value  $\geq 10\%$  of  $Y_p$  to start the process; if this difference then becomes less than 5% of  $Y_p$ , incidence is recorded at that point. If, in addition, the difference changes sign on either side of the incident point, and there exist both a plus difference of absolute value in excess of 10%  $Y_p$ , and a minus difference of

absolute value in excess of 10%  $Y_p$ , then a crossover point  $X_1$ ,  $X_2$ , or  $X_3$  is recorded, depending on whether the incident point was in the upper, mid, or lower third of the previous stroke. Once any crossover is found, the search ends. If a crossover occurs, and a previous point of incidence,  $I$ , was found, the crossover becomes the entry in the final relation code.

The incidence or crossing condition is indicated by a six bit code. These codes are listed in Table 3 of Appendix VI.

#### 4.6 ARCHETYPE CODING

##### 4.6.1 PHILOSOPHY

The recognition method is designed as a fixed rather than adaptive method; and the standard symbols, or archetypes, are coded on the basis of the minimum characteristics needed to identify and distinguish class members. The identifying archetype characteristics can be the specific characteristics detected at input, or more general characteristics defined as some set of specific characteristics. A necessary sequence of the archetype characteristics define the character, usually in a positive way. A dictionary of archetype codes is provided in Appendix VII.

##### 4.6.2 SHAPE CODE MATCHES

All distinct shape codes (those  $\leq 24$ ) must match explicitly, that is, in size, direction, and curvature. These properties are all subsumed by the shape code.

The nondistinct shape codes (those  $\geq 26$ ) are used only in the archetype dictionary. They allow one or two of the three characteristics above to specify a stroke. (For example, any stroke of a given size, or in a given direction.)

Acceptable matches for the distinct and nondistinct shape codes are listed in Table 1 of Appendix VI.

The micro shape code (25) is handled specially. If a micro stroke is specified in the dictionary, it must appear in the input sequence in the corresponding position. A micro stroke in the input sequence which is not present in the dictionary will be ignored.

##### 4.6.3 RETURN AND CLOSURE CODE MATCHES

The relation code matches allow a certain amount of tolerance in the relevant directions. They also allow negative specifications; for example, an open return specification will not accept closure, which is useful in distinguishing "y" from "u". Acceptable matches for the relation codes are listed in Table 2 of Appendix VI.

#### 4.6.4 INCIDENCE AND CROSSING CODE MATCHES

Acceptable matches for the incidence and crossing codes allow variation in both position and degree of incidence. Special codes used only in the dictionary of archetypal stroke sequences allow the condition to occur in the upper or lower thirds of the previous strokes or anywhere on the previous strokes.

When crossing is specified, crossing is required. When incidence is specified, either incidence or crossing will be acceptable. When no condition is specified, incidence may be accepted; crossing will not be accepted. If either incidence or crossing is specified, the no condition situation will not be specified.

These matches are detailed in Table 3 of Appendix VI.

#### 4.7 SOME PROBLEMS WHICH DEVELOPED WITH THE RECOGNITION METHOD AND THEIR SOLUTIONS

##### 4.7.1 NOISE AND SMOOTHING

It was found desirable to have some sort of smoothing of the input point sequence to counteract noise and other irregularities in the input. Noise in the input point sequence may be thought of as jaggedness or lack of smoothness in the point sequence representing a line drawn on the tablet. It has two principle sources: one, the tablet itself; and secondly, the user's hand tremors.

Tablet noise is due to stray pickup from the pen (the pen is capacitively coupled to the raster; any minor electrostatic disturbance can disrupt the information sent back by the pen). It, like the "noise" generated by the user's hand tremor, is essentially unavoidable. However, it can be minimized by maintaining the stylus tip geometry conical, which becomes degraded as a result of wear.

The recognition programs are sensitive to noise, even in small amounts, because of the nature of the stroke segmenting algorithm. One criterion for segmentation is a significant change in the direction of the input point sequence. Vertical noise can result in an essentially horizontal stroke being characterized as a bundle of little up and down strokes. This might be circumvented by tightening up the definition of significant, but there is a limit to how far one can go before losing the required resolution. To get around this difficulty, a smoothing mechanism was incorporated into the input routine. This routine stored points, the coordinates of which were the average of the eight most recent input points from the tablet. The smoothed input was displayed and used as the basis for the stroke segmentation. The noise rejection was excellent, but the averaging process caused the display to seem to lag

the tablet by about four points. This is a consequence of the fact that the coordinates of the averaged point will be roughly midway between the first and last (eighth) points of the average. The averaged point will correspond to the fourth point of the average, which is four points "behind" the current point.

Thus, what the user saw did not actually represent the position of his hand, and the result was undershoot, overshoot, and a general misjudging of how a character was drawn. The smoothing process also very effectively smoothed-out cusps and corners, thus deforming characters with sharp features. This represents a loss of useful information.

One way to get around the eye-hand lag is to display the input point sequence and store the smoothed sequence. This, however, still leaves the problem of smoothed out corners and cusps, and what the user sees does not actually represent the information the recognition programs will use.

A four rather than eight point moving average was then incorporated into the input system with excellent results. The eye-hand lag was thus reduced by half and was then not noticeable.

In addition, the averaging was stopped at a pen lift, and the averaged point sequence was terminated at that point. Originally, averages of the last eight, seven, six, five, etc. points were computed, so that the last points stored approached the last points which came in. A similar process takes place when points first come in so as to move smoothly from the first point into the averaged point sequence as shown below:

OUTPUT POINT	INPUT POINTS IN AVERAGE	NUMBER OF POINTS IN AVERAGE
1	1	1
2	1, 2	2
3	1, 2, 3	3
...	...	...
8	1, ..., 8	8
9	2, ..., 9	8
...	...	...
pen lifted	here, at input point n	...
n-7	n-7, ..., n	8
n-6	n-6, ..., n	7
...	...	...
n-1	n-1, n	2
n	n	1

In the four point average, the initial points are averaged as before, but points are no longer computed and stored after a pen lift. This eliminates the "tail," which suddenly appeared when the pen was lifted in the original system.



#### 4.8 AREAS FOR FUTURE INVESTIGATION

The suggestions to be discussed result primarily from the use of the system during its development. However, none of these suggestions were implemented, since it was felt that studying the basic editing technique was more important than improving the character recognition algorithm.

##### 4.8.1 AN ALTERNATE AVERAGING SCHEME

With a simple moving average, corners and cusps unavoidably lose some of their sharpness. Modifying the averaging technique can minimize this, but at some expense in the smoothness of the output point sequence. A possible modification is to weight the current input point more than the previous points. For example, averaged coordinates might be computed on the basis of the two previous input points as follows:

$$\bar{x}_n = \frac{2x_n + x_{n-1} + x_{n-2}}{4}$$

similarly for  $\bar{y}_n$ , where  $\bar{x}_i, \bar{y}_i$  are the averaged coordinates, and  $x_i, y_i$  are the input coordinates.

This type of average is very easy to implement, and would probably also have proven more efficient with respect to both the execution time and the storage required.

##### 4.8.2 HORIZONTAL SEGMENTING

Experimentation with the recognition programs suggested that it might be useful to define a stroke when a vertically moving sequence begins to move horizontally. The system which was implemented breaks only on vertical direction changes. Thus, to get a horizontal stroke, it has to stand alone or have enough, but not too much vertical motion. A flow chart of the horizontal segmenting algorithm appears in Appendix VIII.

In the proposed horizontal segmenting technique, a point is considered to be moving horizontally if its y change from the previous point is within a small tolerance (for example, + 1 coordinate unit). If a long enough run in one direction is followed by a long enough run in another direction (allowable directions are up, down, and horizontally), a stroke is defined at the direction change. The "long enough" condition is different for horizontal and vertical runs; the horizontal run is required to be longer than the vertical one. An additional feature should be incorporated into this algorithm, that of saving a run's length and direction, and whether it was preceded by a long enough run, whenever a direction change occurred. If the new

run proved not to be long enough, the old run situation would be restored. Thus, small jogs in the point sequence would not disturb an otherwise well established trend.

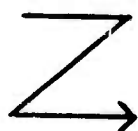
This segmenting algorithm allows for unique stroke specifications for otherwise ambiguous characters. For example, "e" could be specified as follows:



right stroke  
minor upward, +lc  
major downward, -lc  
optional minor upward

In the system implemented, the initial horizontal stroke would not have been defined.

Another example is "z," which could be specified in its easier-to-draw printed configuration:



right stroke  
down stroke  
right stroke

#### 4.8.3 QUADRANT SEGMENTING

An immediate outgrowth of the horizontal segmenting scheme is to consider including horizontal changes of direction in defining a stroke. Thus strokes would be separated into four basic classes, those moving (a) up and to the right; (b) down and to the right; (c) up and to the left; and (d) down and to the left. (See Figure 11.)



Figure 11. Stroke Classifications by Direction

The same criterion of long enough runs and the technique of ignoring too short runs would be used. Curvature and relative size would further characterize each stroke.

Since most interesting corners are acute (and thus at the corner either the x or y direction must change), there is intrinsic corner detection in this segmenting method. A corner is a break between two zero curvature (i.e., straight line) strokes. There is no need to mark a corner as such, because the stroke specifications imply it. Thus, this algorithm would distinguish between the following characters:



The first is two straight strokes ("d" and "b" as denoted above), the second is two curved strokes.

There would be no need for the type 2 curvature, as most inflection points would also be stroke break points.

An obvious difficulty with this technique is that the specifications are no longer concise. For example, the "(" character is now two strokes instead of a single stroke. This is somewhat redundant -- a single stroke would quite adequately specify the character. In the case of "<", however, we want two strokes because there are essentially two distinct strokes. A possible solution is to add a postprocessor to the stroke segmenter, which would examine pairs of successive strokes moving in the same vertical direction, and if they have a continuous curvature, would combine them. Thus, a circle drawn as follows:



would first be broken into four strokes with the same curvature:



and then the downward pair and the upward pair would be combined into single strokes.

This segmenting technique would allow the recognition of both printed and cursive handwriting, as well as a great variety of special symbols. Relation coding would of course be retained, since it provides extremely useful information about the character.

#### 4.8.4 REORDERING OF THE DICTIONARY

The dictionary is presently ordered in a simple alphabetic-like manner. This has proven acceptable, as recognition times are well within operator reaction times. If, however, an extremely large dictionary were found to be necessary or even useful, some means of speeding up the search for the archetype would be imperative. There are two alternatives -- a probabilistic ordering of the archetypes, or a list structuring.

##### 4.8.4.1 PROBABILISTIC ORDERING

This method is the simplest and most easily accomplished. The archetypes are simply ordered with respect to the frequency of occurrences of the symbols they represent -- most frequent first, least frequent last. This would require no reprogramming and should produce a significant improvement in search time.











#### 4.8.4.2 LIST STRUCTURING

At significant analytic and reprogramming cost, the dictionary could be list structured. A cursory examination of this approach has shown the difficulties to be formidable, with the result of reducing the search time to almost the absolute minimum. The list would be only as deep as the largest number of strokes in a single character. The span of the list would be rather large, since a new node at the next level down is required for each stroke, which might possibly (with respect to the set of characters to be recognized) follow the stroke represented by the current node. The overhead involved in maintaining the list, in terms of the space requirement, would probably override the usefulness of the improvement in search time.

#### 4.8.5 AN ALTERNATE RECOGNITION METHOD: RECOGNITION BY MAJOR SUB-FEATURES

Character recognition by major sub-features is based on breaking up the characters into often-recurring topologically distinguishable sub-features. This is somewhat similar to the sub-stroke technique now used, but on a cruder level of analysis.

It was found that all characters except "r" could be construed from concatenations of 11 highly distinguishable character parts. These character parts are as follows:

NAME	SHAPE	DESCRIPTION
B		2 concatenated tails (see T below)
C		A lead stroke, short counter-clockwise open loop, horizontally oriented
E		Short counter-clockwise loop, with a tail upward (i.e., above the baseline)
F		Long counter-clockwise loop, with a tail downward
J		Long clockwise loop, downward
L		Long counter-clockwise loop, no tail, upward
N		Clockwise, concave downward
S		Short clockwise loop, with or without a cusp, with a tail
T (tail)		Counter-clockwise, concave upward
X (cross)- or /		Linear stroke
D (dot)		self-explanatory



The lower case script alphabet is constructed from these parts as follows. □ indicates a pen lift. Parentheses indicate an optional stroke.

CHARACTER	STROKE SEGMENTATION	STROKE DESIGNATION
A	<i>c l</i>	CE
B	<i>l v</i>	LB
C	<i>c</i>	C
D	<i>c h</i>	CLT
E	<i>e</i>	E
F	<i>f</i>	LF
G	<i>c j</i>	CJ
H	<i>h n</i>	LNT
I	<i>i</i>	E□D
J	<i>j</i>	TJ□D
K	<i>k s</i>	LS
L	<i>l</i>	LT
M	<i>m m m</i>	NNNT
N	<i>n n</i>	NNT
O	<i>o</i>	CB
P	<i>p s</i>	TJS
Q	<i>c q</i>	CEF
R	<i>r</i>	
S	<i>s</i>	(T)S
T	<i>t</i>	E□X
U	<i>u u</i>	EE
V	<i>v</i>	NB
W	<i>w w w</i>	EEE
X	<i>x</i>	NT□X

CHARACTER	STROKE SEGMENTATION	STROKE DESIGNATION
Y		EJ
Z		NNJ

Recognition of and segmentation into subparts would be carried out in much the same manner as the current system identifies strokes. This representation is more compact than the current one, and the subparts are sufficiently distinct so as to allow liberal variations to accommodate different users' handwriting.

## THE EDITING CONSOLE

Figure 12 illustrates the editing console designed. This console incorporates a RAND Tablet, storage-type CRT display unit, input keyboard and working surfaces. In this design, an attempt is made to ideally arrange these different components in order to enable efficient editing.



Figure 12. The Editing Console  
(Arrangement for a Right-Handed User)

The console employs a modular design concept, whereby, the console layout could be easily modified (rearranged) in order to provide an optimal working arrangement. The console consists of three tables, two of which are interchangeable, and a cart (on wheels) for the display unit, which can be oriented for improved visibility. On one of the interchangeable tables is located the input keyboard, and the second one provides additional workspace. The RAND Tablet is located on the third table.

The optimal arrangement for a right-handed user was found to consist of the table with the keyboard to the left of the table with the RAND Tablet, and the other table on the right. Thus, the user could easily use either the keyboard or the tablet. For a left-handed user, the optimal arrangement was found to be a mirror image of that for the right-handed user.

## VI

### CONCLUSIONS

The MOSAIC system provides a means for performing computer-aided text editing in a much more natural fashion than the technique employing a CRT display and a light pen. The use of the RAND Tablet stylus is very similar to using a pencil. The feasibility of the use of a RAND Tablet in text editing relies heavily on the effectiveness of the character recognition scheme. In the MOSAIC System, the recognition of cursive writing is performed by using geometric-topological invariants of stroke succession. This method was found to have great potential, in that, most of the characters drawn on the tablet were recognized correctly. Suggested improvements to the recognition algorithm have been provided in the text of this report. (See Section 4.8.)

It is felt that if someone were to use the RAND Tablet for text editing purposes today, he would use it in the manner in which it is used in the MOSAIC System. A summary of the tentative conclusions resulting from informal observation and use of the system during its development will now be presented.

#### 6.1 THE RAND TABLET

The capabilities of the RAND Tablet in editing are quite extensive, and have been only partially explored. Its use in the editing of linear text has been considerable, due in large part to the directness of the x-y location. It is felt that an extension to more complicated scientific data where two-dimensional reorientation is also required would show even greater advantage. The principle problem found in using the tablet was its fragility. During the development of the system, the pen tip had to be replaced many times. This would be very undesirable in a production environment.

#### 6.2 SYMBOLIC CONTROL: A BASIC TABLET TECHNIQUE

Control of computer operations by on-line handwriting, via the RAND Tablet, has two compelling advantages with respect to alternative typewriter methods. The user is free from the constraints of linearity and from the restrictions of symbol sets of small number and uniform size. Hence, it is now possible to make use of hand-drawn alphabets of arbitrary symbols as command inputs of great simplicity. When these operators can be directly associated with their operands, as in the case of making an editing mark on a word, control efficiency is high and errors are few. In other cases, the position of the marks with respect to the data or to other marks can serve as command determinatives to permit the extensive modification of the intent of these commands in simple ways. In text editing, for instance, the specification of foreign alphabetic symbols is possible in several

optional ways: direct drawing, drawing of the English transliteration in a determinative area, or template-pointing to one element of a select class.

We feel the extension of symbolic control methods in the manipulation of text will prove it to be a basic tablet technique having wide applications.

### 6.3 THE STORAGE-TYPE CRT DISPLAY UNIT

The storage-type CRT display unit was found to be particularly well-suited to text editing. The text need only be displayed once in the storage mode, and then erased and redisplayed only when changes are made to the text, as opposed to the continuous refreshing of the display, as required by non-storage-type units. This feature is very valuable in a multi-console on-line editing environment, since the computer time necessary for maintaining each display unit is minimized. The principle disadvantage found in using the storage-type CRT display unit was the irritation caused by the flashes resulting from erasing the display. However, this irritation was found to be less than that caused by the constant flicker of a non-storage-type unit.

### 6.4 THE EDITING CONSOLE

The editing console design seemed to enable efficient editing. The principle features of this console were found to be that the modularity in construction provided the ability to easily rearrange the units for the user's convenience, sufficient work areas were available, and the visibility of the display unit was improved by the cart on which it was placed.

**BLANK PAGE**

## APPENDIX I

### MOSAIC-OPERATION AND COMMANDS

#### 1. PROGRAM START-UP

There are two fixed starting locations:

Location 4 - Start; clear all three buffers, reset number of lines displayed and first line displayed, etc.

Location 5 - Restart; all parameters and buffer contents are left as is.

#### 2. SENSE SWITCH OPTIONS

SS $\downarrow$  down - Normal display.

SS $\uparrow$  up - Non-printing characters are displayed as follows:  
spaces  $\square$ , tabs  $\rightarrow$ , carriage returns  $\hookrightarrow$ .

Sense switch  $\uparrow$  can be altered at any time, and the change will be reflected on the next display. The sense switch effects only the text buffer display. Non-printing characters are always displayed in the record delimiter and search buffer displays.

#### 3. INPUT KEYBOARD COMMANDS

All of the following commands are followed by a RETURN character. If n (any number) is not specified, it is assumed to be "1."

##### 3.1 PAPER TAPE INPUT COMMANDS

- IN Clear text buffer and read in text until a stopcode is reached.
- nQ Quantized read. Clear text buffer and read in text until n record delimiters are reached. Delimiters are set by the "SQ" command.
- nAP Append records to text buffer until n delimiters are reached.
- DI Read in and store a new tablet recognition dictionary tape.

### 3.2 TABLET PEN COMMANDS

PE Activate the pen. The real-time clock is turned on.

### 3.3 PAPER TAPE OUTPUT COMMANDS

P Punch text buffer contents, followed by a stopcode and feed (a section of blank tape).

PF Punch text buffer contents and feed (no stopcode).

PT Punch text buffer contents only.

SC Punch a stopcode and feed only.

FD Punch feed only.

### 3.4 COMBINED PAPER TAPE INPUT/OUTPUT COMMANDS

nTT Read and punch records until n delimiters are reached.

PN Combination of a "PT" and an "IN" command.

PQ Combination of a "PT" and a "Q" command.

### 3.5 CRT DISPLAY CONTROL COMMANDS

nCK Change the delete/change command timer to n display times. (See Appendix I, Section 4.1)

nR Reset first line to be displayed to line n.

nN Set number of lines to be displayed to n (if greater than 24 maximum, will be set to 24).

nA Move display ahead n lines.

nB Move display back n lines.

RETURN (only) Initiate a new display.

### 3.6 RECORD DELIMITER SET-UP AND SEARCH COMMANDS

SQ Set quantized read delimiter(s). Text Mode is entered, and the user types any sequence of characters, which delimit a logical record. Return to Control Mode by typing two sequential case shifts.

nF Find string. Text Mode is entered and the string of characters to be searched for (starting at line n) is



typed in, terminated by two case shifts. The first line containing the string is brought to the top of the text buffer display. (If not found, the last text line in the buffer is displayed.)

- nFC Continue searching for the next occurrence of the string presently in the search buffer. If n is not specified, start from last occurrence.
- nFP Find parity error. Same as "F," except the program automatically sets up a single parity error character in the search buffer. Use "FC" to continue parity error search.

#### 4. TABLET COMMANDS

By drawing over fixed control areas on the tablet, regardless of the character drawn, the following operations can be performed. (See Figure 13.) Numerics can be drawn and accumulated prior to touching the control area. These will be used as the arguments for applicable operations.

- Control area 1 - Move display back n lines (same as keyboard command "B").
- Control area 2 - End of text. Complete the editing function. Return to Control Mode.
- Control area 3 - Quantized paper tape read (same as "Q").
- Control area 4 - Reset first line displayed (same as "R").
- Control area 5 - Equivalent to a carriage return (re-displays when in Control Mode).
- Control area 6 - Move display ahead n lines (same as "A").
- Control area 7 - Equivalent to a SPACE.
- Control area 8 - Punch paper tape (same as "P").

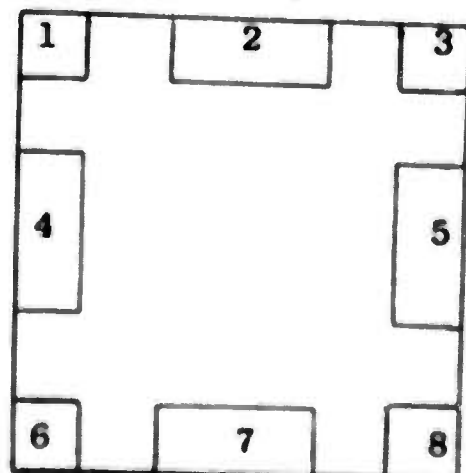


Figure 13. The Tablet Control Areas

#### 4.1 CONTROL MODE EDIT COMMANDS

The following symbols must be drawn on the tablet, over the position corresponding to the display, where the editing function is to take place. The arrows indicate drawing direction only and are not part of the symbols.

↓ Select a  
Character

A single downward stroke drawn through a character has two functions: change and delete. As soon as the select symbol is detected, a shift to Text Mode is effected, and any alpha- numerics drawn anywhere on the tablet are taken to be substitutions for the selected symbol. If, after drawing the select symbol, no additional data is drawn in, it is assumed that a delete is intended, the symbol selected is deleted, and the program returns to Control Mode. The time period in which entry must take place is a variable, which can be set by the "CK" keyboard command.

↪ Change Word

Drawing a "w" through a word indicates "Change Word." This function is analogous to the "Select Character" function above, except that an entire word is selected for change. In addition, no time-based switch for deletion is available. The previous characters in the word are replaced by spaces.

↪ Change Line

Same as "Change Word," except the complete line on which the "l" is drawn, is selected and replaced by spaces.

→ Delete Word(s)

A single left-to-right stroke drawn through a word, group of words, or line, deletes that data and returns to Control Mode.

X Delete Line

An "x" drawn anywhere on a line will delete the entire line.

↪ Transpose

When this symbol is drawn through two words, the words will be transposed. When this symbol is drawn through a single word, it will transpose the letter that the last up-stroke impinges on and the one immediately to the left of it.

d Less Caps

Drawing a "d" (decapitalize) through a word causes the capitalization to be reduced by one level. The three decreasing levels of capitalization are as follows:

## CAPITALS

Capitals

capitals

 More Caps

Drawing a "c" through a word causes the capitalization to be increased by one level. It is the inverse of the "d" command, above.

 Insert

The insert mark points at the character immediately to the right of the point at which insertion is to take place. When this symbol is detected, the program switches to Text Mode and accepts either hand-drawn alphanumerics or keyboard input, which will be inserted at the selected spot.

 Display

The drawing of a carriage return will initiate a new display when in Control Mode.

## 4.2 TEXT MODE CHARACTER INSERTION

Text Mode is entered via the insert or select commands, and alphanumerics may be entered at any place on the tablet. Several classes of symbols may be entered in Text Mode.

- (a) **Alphabetic Characters.** The lower case cursive alphabet is used, according to the standard Palmer method of penmanship, with modifications to allow for ease and speed of writing. A moderate degree of variation is acceptable.
- (b) **Numeric Characters.** The digits zero through nine are drawn according to basic Palmer standards. Exceptions are the zero, which is slashed (0), and the one, which contains a top hook and flat base (1).
- (c) **Non-Printing Symbols.** Again, arrowheads show stroke direction, but are not part of the symbols.

 Carriage Return (or use control area 5)

→ Space (or use control area 7)

← Backspace (deletes previous symbol)

↑ Upshift

↓ Downshift

→↓ Tab

(d) Punctuation Marks.

•↓ Period

—→ Middle Dot (non-spacing)

•↓ Comma

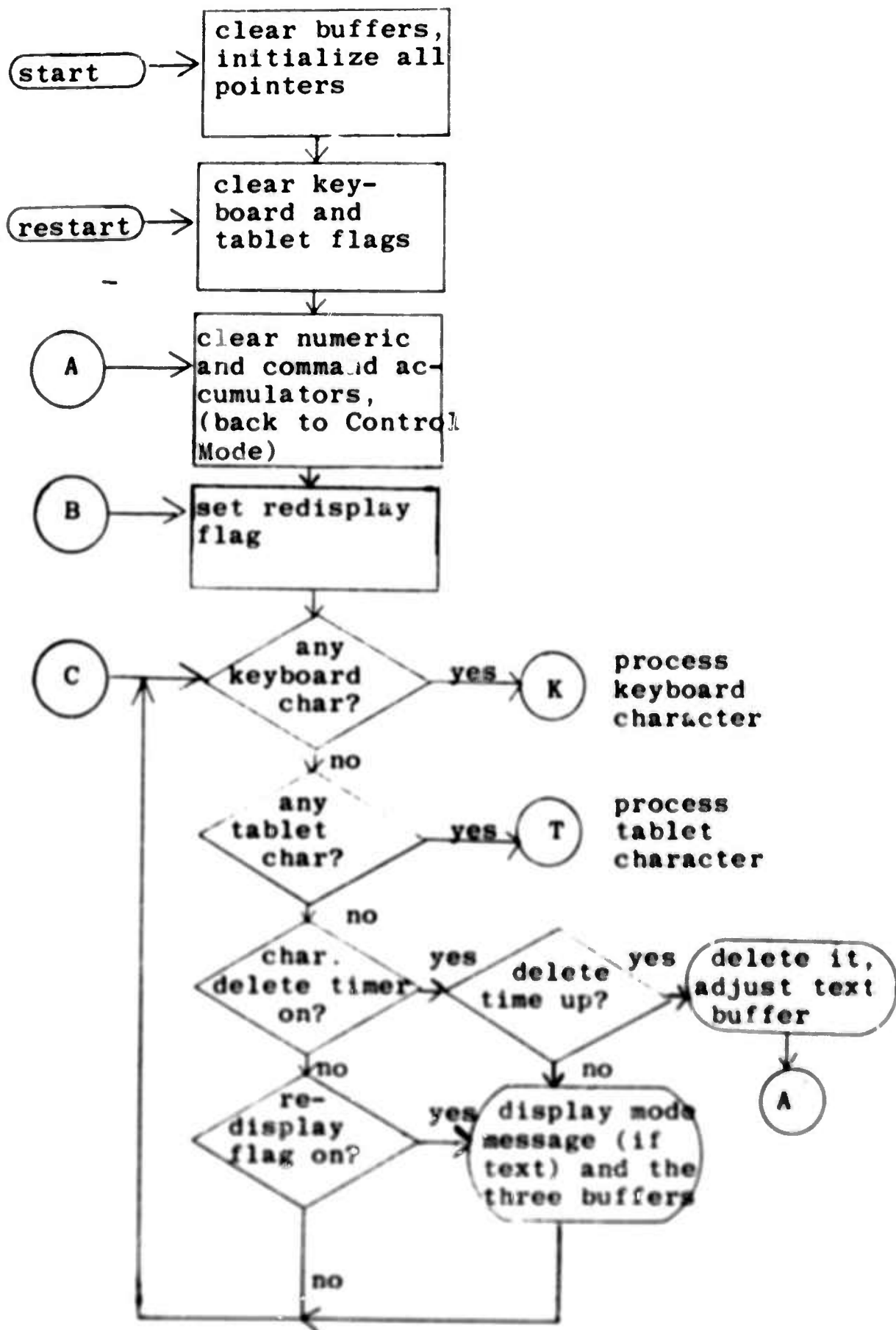
The above set used in combination can also produce colons and semicolons.

#### 4.3 EXITING TEXT MODE

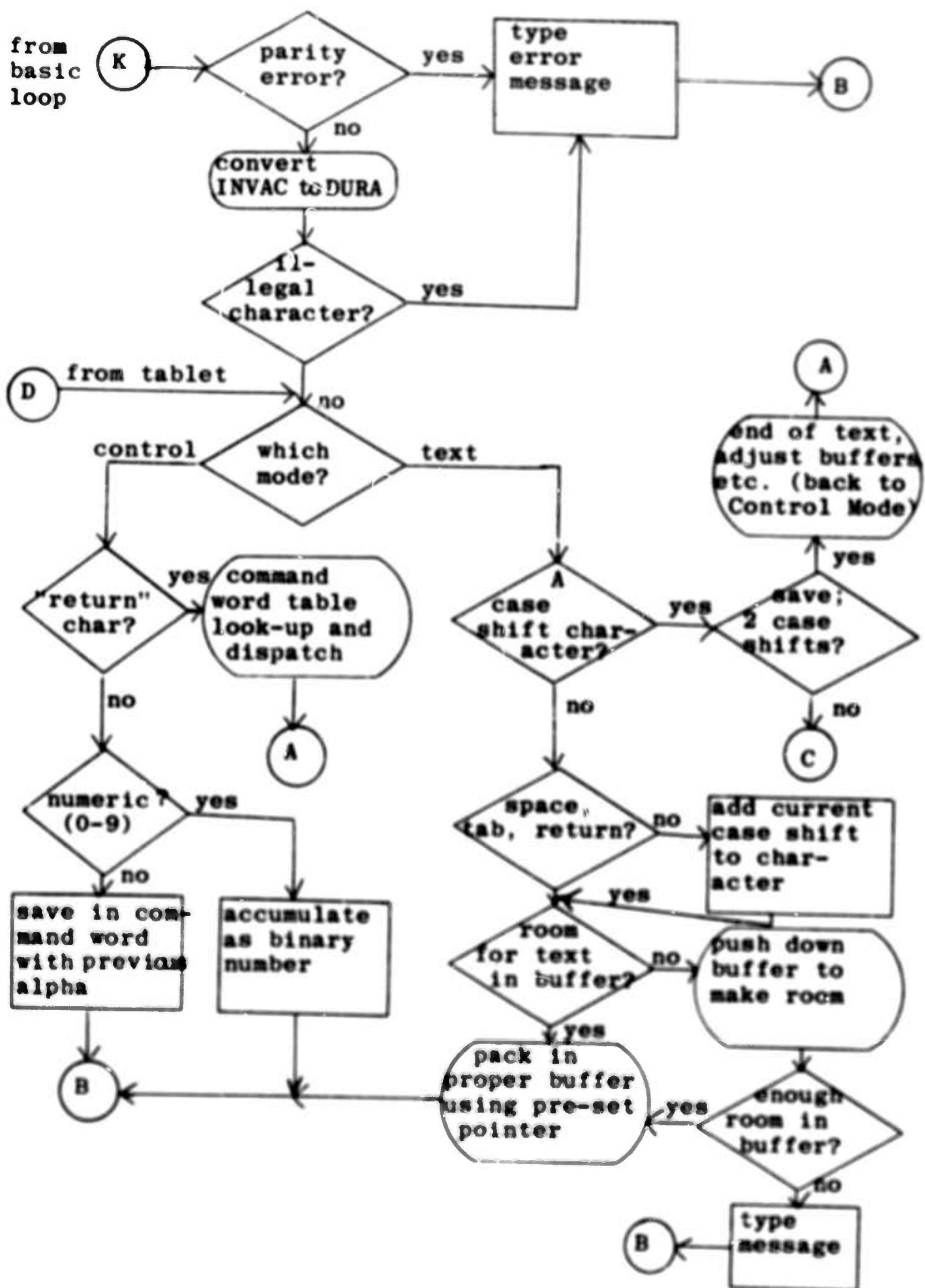
Note that ending insertion of text and returning to Control Mode may be done in three ways:

- (a) Typing two sequential case shifts (any combination of upper and lower case) on the keyboard.
- (b) Drawing two sequential case shifts on the tablet.
- (c) Striking the end of text control area on the tablet (top middle-2).

# APPENDIX II

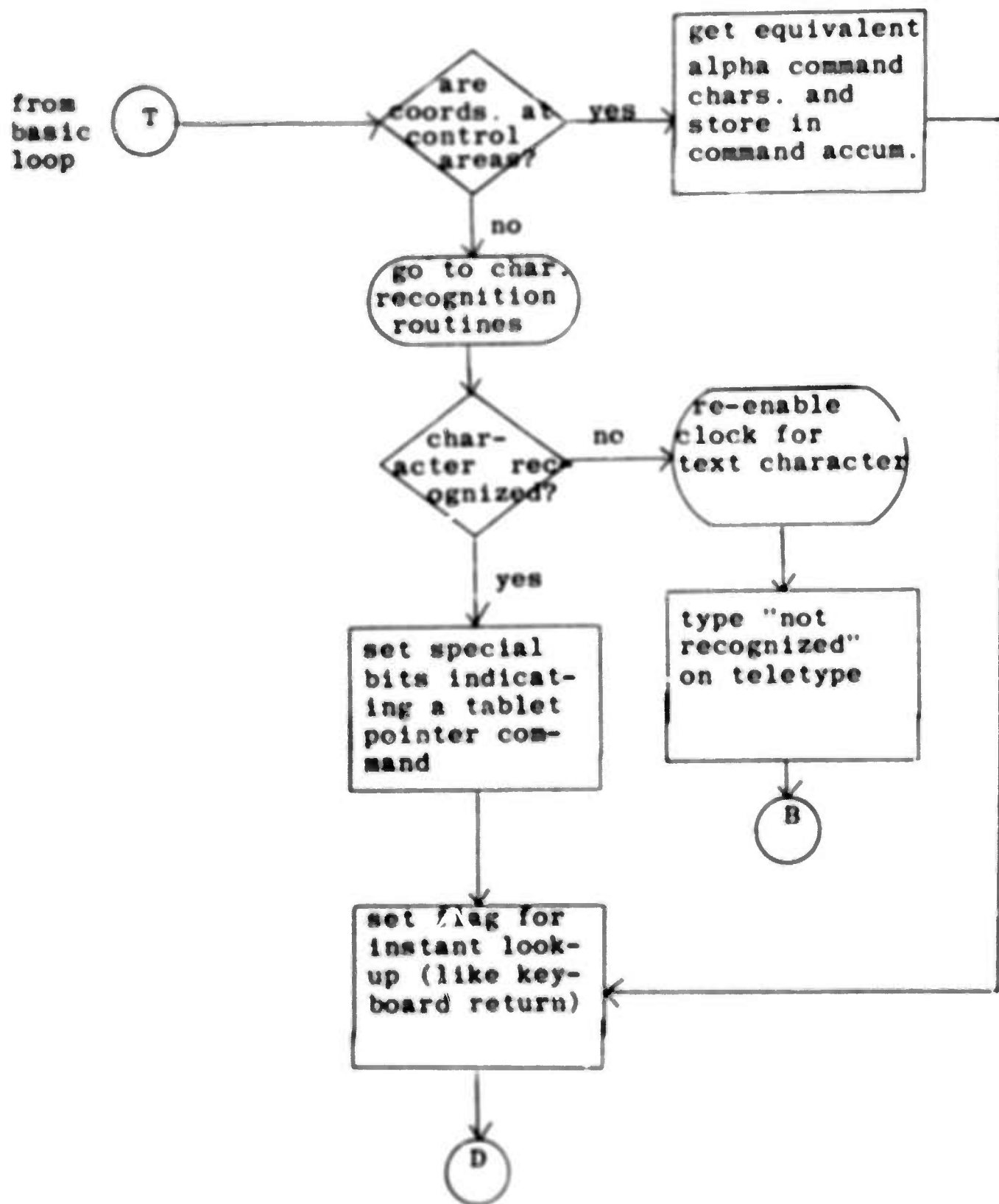


FUNCTIONAL FLOW CHART OF TEXT EDITOR



FUNCTIONAL FLOW CHART OF TEXT EDITOR (Continued)



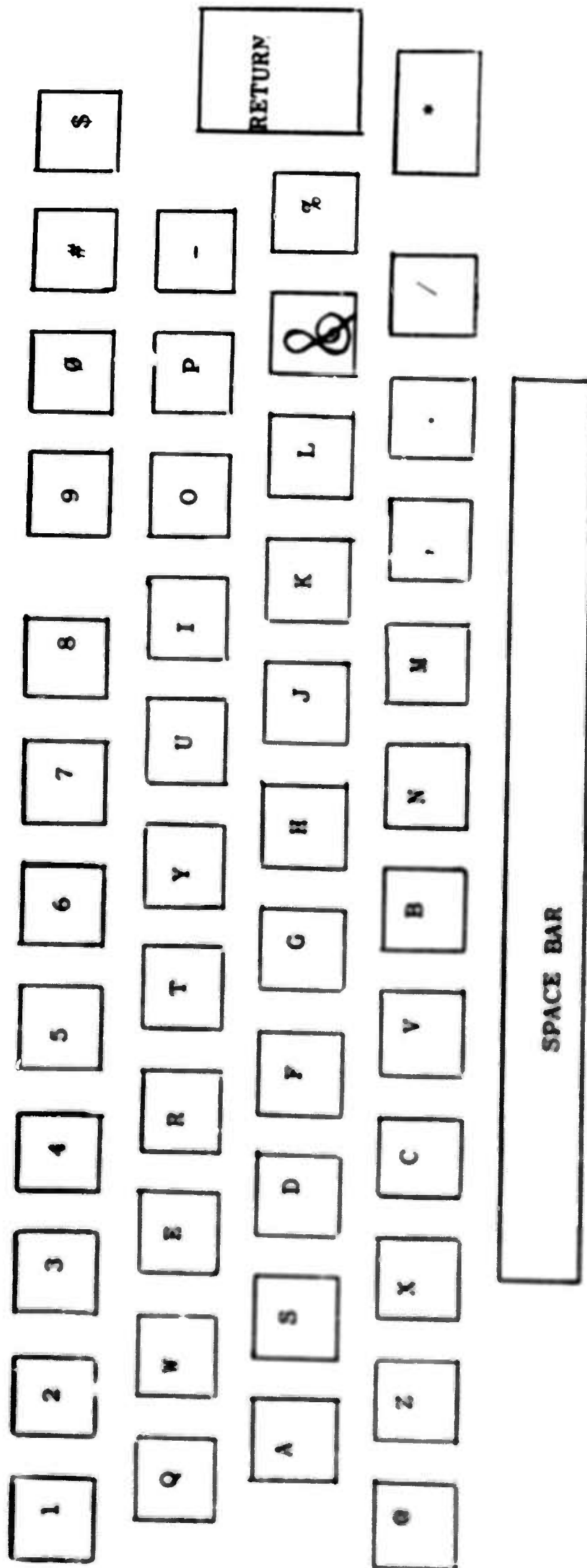


FUNCTIONAL FLOW CHART OF TEXT EDITOR (Continued)

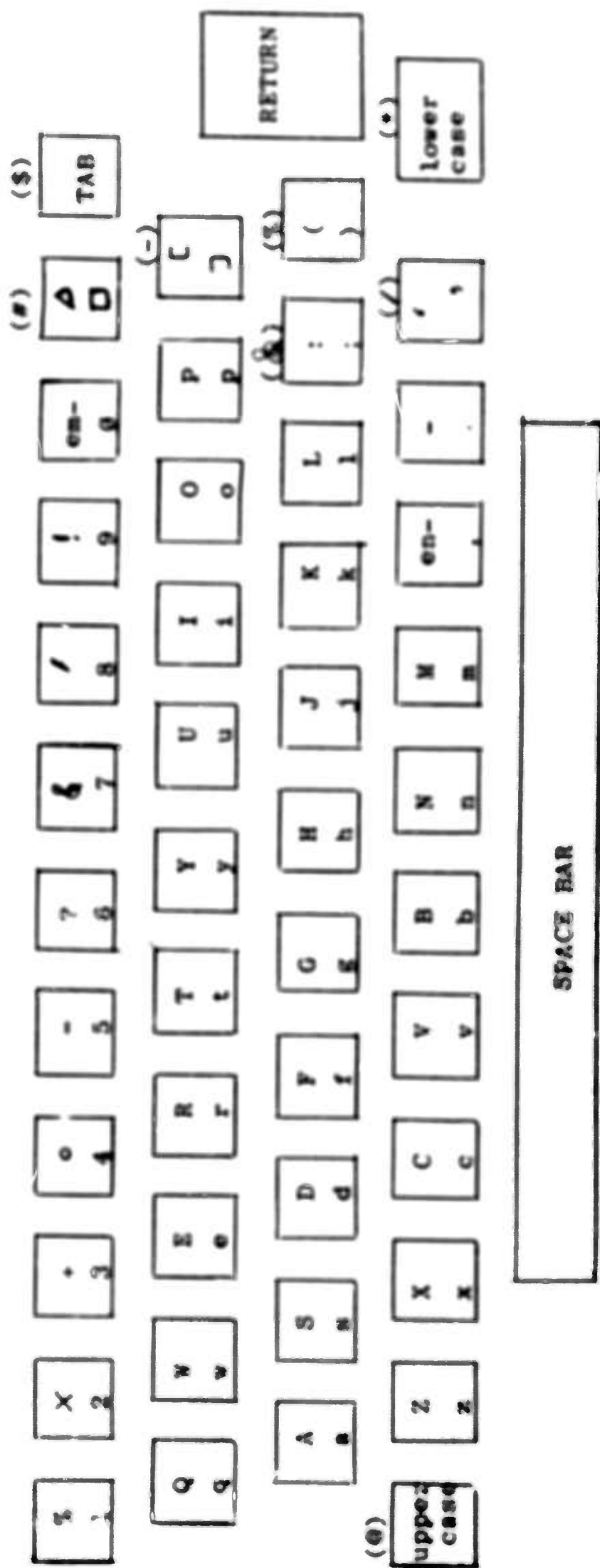
**APPENDIX III**  
**LAYOUT OF THE INPUT KEYBOARD**



# STANDARD INVAC KEYBOARD



# INVAC KEYBOARD SIMULATION OF DURA KEYBOARD



NOTES: (1) Characters in parentheses above a key indicate its actual marking when different from its simulated marking.

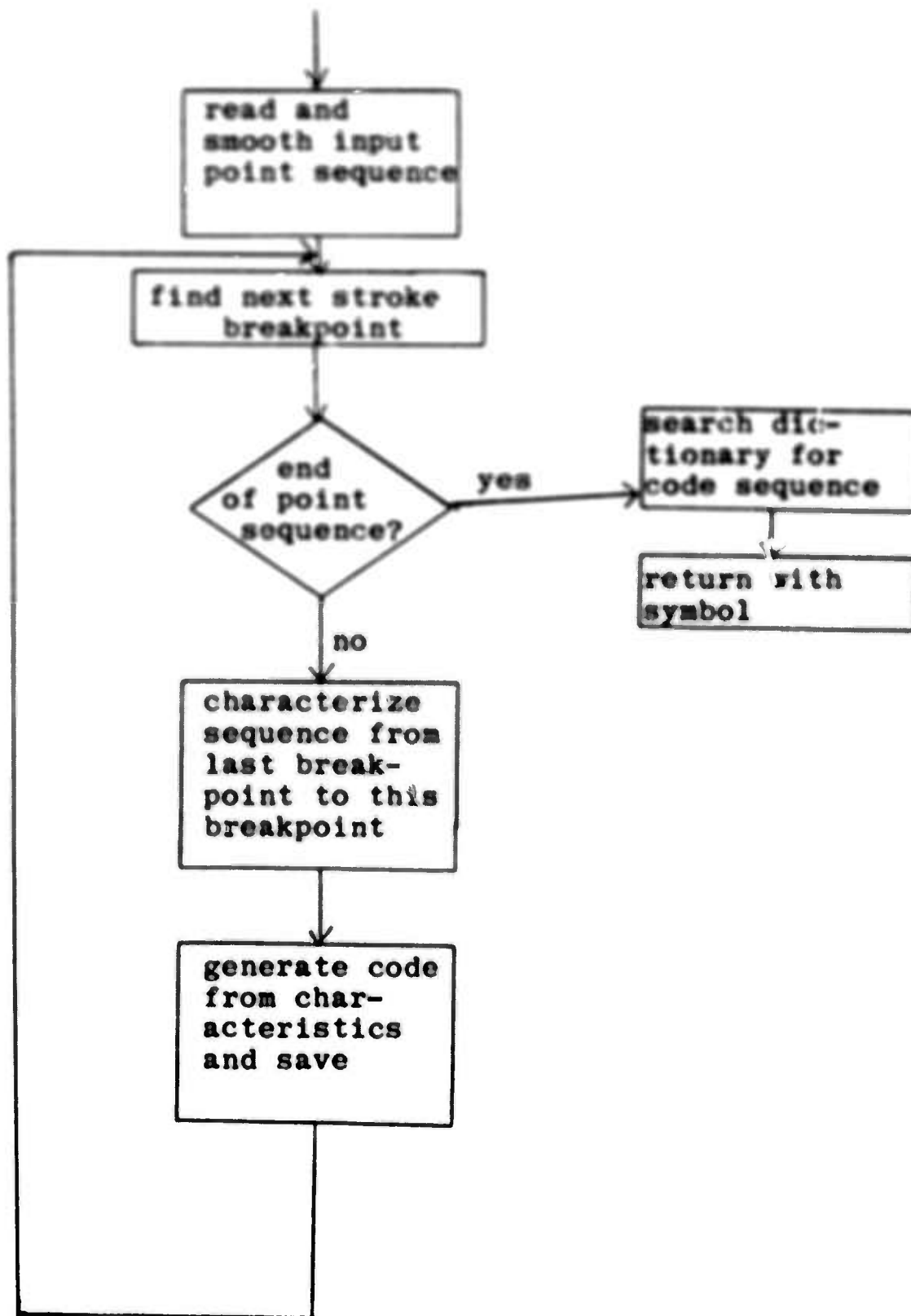
(2) Case is ignored when user is in "Command Mode."

#### APPENDIX IV

##### THE TELETYPE MESSAGES

MOSAIC	The MOSAIC System has just been initialized.
B	The input keyboard is busy. Retype the last character.
KE?	Keyboard error. An illegal character was typed.
UC?	Unknown command. The sequence of input characters typed before the RETURN did not constitute a legal command.
NR?	Not recognized. The tablet character drawn could not be interpreted by the recognition routines.
PT?	Pointer. An editing command was drawn, which was not pointing to an area of the CRT containing text.
Pnnn	Parity error. A character nnn (octal) was read from paper tape, which was illegal or bad (even) parity.
FUL	The text buffer is nearly full. Paper tape input ceases.
FUL1	The text buffer is completely full. No more characters may be inserted.

## APPENDIX V



FUNCTIONAL FLOW CHART OF RECOGNITION ALGORITHM

# APPENDIX VI

## STROKE CODES

### Distinct Codes

<u>Symbolic</u>	<u>Description</u>	<u>5 Bit Octal Code</u>
	pen lift	0
↓	major, -y, 0c	1
↑	major, +y, 0c	2
↙	major, -y, -lc	3
↘	major, +y, -lc	4
↖	major, -y, +lc	5
↗	major, +y, +lc	6
↙↙	major, -y, -2c	7
↘↘	major, +y, -2c	10
↖↖	major, -y, +2c	11
↗↗	major, +y, +2c	12
←	major, -x	13
→	major, +x	14
←	minor, -x	15
→	minor, +x	16
↓	minor, -y, 0c	17
↑	minor, +y, 0c	20
↙	minor, -y, -lc	21
↘	minor, +y, -lc	22
↖	minor, -y, +lc	23
↗	minor, +y, +lc	24
	micro	25

Table 1. Shape Codes

<u>Nondistinct Codes</u>		
<u>Description</u>	<u>5 Bit Octal Code</u>	<u>Included Shape Codes (Acceptable Matches)</u>
major stroke	26	all major codes, (1 through 14)
minor stroke	27	all minor codes (15 through 24)
major upward	30	all major +y codes (2,4,6,10,12)
minor upward	31	all minor +y codes (20,22,24)
major downward	32	all major -y codes (1,3,5,7,11)
minor downward	33	all minor -y codes (17,21,23)
upstroke	34	all +y codes (2,4,6,10,12,20 22,24)
downstroke	35	all -y codes (1,3,5,7,11,17 21,23)
right stroke	36	all +x codes (14,16)
left stroke	37	all -x codes (13,15)

Table 1. Shape Codes (Continued)

<u>Condition</u>	<u>Symbolic</u>	<u>2 Bit Octal Code</u>	<u>Octal Value In Code Word</u>	<u>Acceptable Match</u>
none	O	0	0	any condition
return	R	1	40	return or closure
closure	C	2	100	closure
open return	U	3	140	return or open return

Table 2. Return and Closure

<u>Condition</u>	<u>Symbolic</u>	<u>6 Bit Octal Code</u>	<u>Octal Value In Code Word</u>	<u>Acceptable Match</u>
top crossing	$X_1$	1	200	$X_1$
middle crossing	$X_2$	2	400	$X_2$
bottom crossing	$X_3$	4	1000	$X_3$
upper crossing	$X_{12}$	3	600	$X_1$ or $X_2$ (archetype only)
lower crossing	$X_{23}$	6	1400	$X_2$ or $X_3$ (archetype only)
crossing	$X$	7	1600	$X_1$ or $X_2$ or $X_3$ (archetype only)
top incidence	$I_1$	11	2200	$X_1$ or $I_1$
middle incidence	$I_2$	22	4400	$X_2$ or $I_2$
bottom incidence	$I_3$	44	11000	$X_3$ or $I_3$
upper incidence	$I_{12}$	33	6600	$X_1$ or $X_2$ or $I_1$ or $I_2$ (archetype only)
lower incidence	$I_{23}$	66	15400	$X_2$ or $X_3$ or $I_2$ or $I_3$ (archetype only)
incidence	$I$	77	17600	$X_1$ or $X_2$ or $X_3$ or $I_1$ or $I_2$ or $I_3$ (archetype only)
no incidence or crossing	0	0	0	no incidence or crossing, or $I_1$ or $I_2$ or $I_3$

Table 3. Incidence and Crossing Codes



# APPENDIX VII

## DICTIONARY ARCHETYPE CODES

SYMBOL	OCTAL CODES	STROKE ARCHETYPES	
A	40027	MINOR STROKE	(OPTIONAL)
	3	MAJOR, -Y, -1C	
	130	MAJOR UPWARD, C	
	75	DOWNSTROKE, R	
	40027	MINOR STROKE	(OPTIONAL)
	20041	END 41	
A	40031	MINOR UPWARD	(OPTIONAL)
	3	MAJOR, -Y, -1C	
	46	MAJOR, +Y, +1C, R	
	143	MAJOR, -Y, -1C, U	
	40031	MINOR UPWARD	(OPTIONAL)
	20041	END 41	
B	6	MAJOR, +Y, +1C	
	1472	MAJOR DOWNWARD, R, X23	
	22	MINOR, +Y, -1C	
	20042	END 42	
B	6	MAJOR, +Y, +1C	
	1472	MAJOR DOWNWARD, R, X23	
	31	MINOR UPWARD	
	16	MINOR, +X	
	20042	END 42	
B	6	MAJOR, +Y, +1C	
	1543	MAJOR, -Y, -1C, U, X23	
	24	MINOR, +Y, +1C	
	36	RIGHT STROKE	
	20042	END 42	
C	40037	LEFT STROKE	(OPTIONAL)
	3	MAJOR, -Y, -1C	
	40027	MINOR STROKE	(OPTIONAL)
	20063	END 63	

24 MINOR, +Y, +IC  
 3 MAJOR, -Y, -IC  
 24 MINOR, +Y, +IC  
 20063 END 63

D

40027 MINOR STROKE (OPTIONAL)  
 21 MINOR, -Y, -IC  
 30 MAJOR UPWARD  
 72 MAJOR DOWNWARD, R  
 40027 MINOR STROKE (OPTIONAL)  
 20044 END 44

D

40031 MINOR UPWARD (OPTIONAL)  
 21 MINOR, -Y, -IC  
 6 MAJOR, +Y, +IC  
 1143 MAJOR, -Y, -IC, U, X3  
 40031 MINOR UPWARD (OPTIONAL)  
 20044 END 44

E

36 RIGHT STROKE  
 31 MINOR UPWARD  
 32 MAJOR DOWNWARD  
 40027 MINOR STROKE (OPTIONAL)  
 20065 END 65

E

30 MAJOR UPWARD  
 1472 MAJOR DOWNWARD, R, X23  
 20065 END 65

E

36 RIGHT STROKE  
 15 MINOR, -X  
 32 MAJOR DOWNWARD  
 40027 MINOR STROKE (OPTIONAL)  
 20065 END 65

E

36 RIGHT STROKE  
 24 MINOR, +Y, +IC  
 3 MAJOR, -Y, -IC  
 40024 MINOR, +Y, +IC (OPTIONAL)  
 20065 END 65

F

24 MINOR, +Y, +IC  
 1632 MAJOR DOWNWARD X  
 17634 UPSTROKE I  
 40027 MINOR STROKE (OPTIONAL)  
 20066 END 66

F

24	MINOR, +Y, +IC	
1003	MAJOR, -Y, -IC	X3
24	MINOR, +Y, +IC	
36	NIGHT STROKE	
20066	END 66	

G

40037	LEFT STROKE	(OPTIONAL)
21	MINOR, -Y, -IC	
131	MINOR UPWARD, C	
32	MAJOR DOWNWARD	
1622	MINOR, +Y, -IC	X
20047	END 47	

G

40037	LEFT STROKE	(OPTIONAL)
21	MINOR, -Y, -IC	
131	MINOR UPWARD, C	
32	MAJOR DOWNWARD	
1604	MAJOR, +Y, -IC	X
20047	END 47	

H

6	MAJOR, +Y, +IC	
1472	MAJOR DOWNWARD, R, X23	
31	MINOR UPWARD	
173	MINOR DOWNWARD, U	
40027	MINOR STROKE	(OPTIONAL)
20050	END 50	

I

30	MAJOR UPWARD	
172	MAJOR DOWNWARD, U	
40027	MINOR STROKE	(OPTIONAL)
0	PEN LIFT	
25	MICRO	
20071	END 71	

J

40027	MINOR STROKE	(OPTIONAL)
32	MAJOR DOWNWARD	
1634	UPSTROKE X	
0	PEN LIFT	
25	MICRO	
20021	END 21	

24 MINOR, +Y, +IC  
 1632 MAJOR DOWNWARD A  
 31 MINOR UPWARD  
 33 MINOR DOWNWARD  
 20022 END 22

L

6 MAJOR, +Y, +IC  
 1472 MAJOR DOWNWARD, A, X23  
 24 MINOR, +Y, +IC  
 20003 END 3

M

4 MAJOR, +Y, -IC  
 72 MAJOR DOWNWARD  
 44 MAJOR, +Y, -IC, R  
 72 MAJOR DOWNWARD, U  
 44 MAJOR, +Y, -IC, R  
 72 MINOR STROKE (OPTIONAL)  
 20024 END 24

N

4 MAJOR, +Y, -IC  
 32 MAJOR DOWNWARD  
 44 MAJOR, +Y, -IC, R  
 72 MAJOR DOWNWARD, R  
 40027 MINOR STROKE (OPTIONAL)  
 20005 END 5

O

3 MAJOR, -Y, -IC  
 186 MAJOR, +Y, +IC, C  
 40027 MINOR STROKE (OPTIONAL)  
 20006 END 6

O

37 LEFT STROKE  
 3 MAJOR, -Y, -IC  
 146 MAJOR, +Y, +IC, U  
 40027 MINOR STROKE (OPTIONAL)  
 20006 END 6

P

40024 MINOR, +Y, +IC (OPTIONAL)  
 1 MAJOR, -Y, -IC  
 38 MAJOR UPWARD  
 23 MINOR, -Y, +IC  
 40027 MINOR STROKE (OPTIONAL)  
 20027 END 27

40027 MINOR STROKE (OPTIONAL)  
 21 MINOR, -Y, -IC  
 131 MINOR UPWARD, C  
 32 MAJOR DOWNWARD  
 17624 MINOR, +Y, +IC I  
 40027 MINOR STROKE (OPTIONAL)  
 20030 END 30

Q  
 40027 MINOR STROKE (OPTIONAL)  
 21 MINOR, -Y, -IC  
 131 MINOR UPWARD, C  
 32 MAJOR DOWNWARD  
 17636 MAJOR, +Y, +IC I  
 40027 MINOR STROKE (OPTIONAL)  
 20030 END 30

R  
 6 MAJOR, +Y, +IC  
 33 MINOR DOWNWARD  
 40033 MINOR DOWNWARD (OPTIONAL)  
 33 MINOR DOWNWARD  
 40027 MINOR STROKE (OPTIONAL)  
 20011 END 11

R  
 6 MAJOR, +Y, +IC  
 33 MINOR DOWNWARD  
 36 RIGHT STROKE  
 33 MINOR DOWNWARD  
 40027 MINOR STROKE (OPTIONAL)  
 20011 END 11

S  
 6 MAJOR, +Y, +IC  
 45 MAJOR, -Y, +IC, R  
 40027 MINOR STROKE (OPTIONAL)  
 20062 END 62

T  
 30 MAJOR UPWARD  
 172 MAJOR DOWNWARD, U  
 40027 MINOR STROKE (OPTIONAL)  
 8 PEN LIFT  
 36 RIGHT STROKE  
 20043 END 43

U  
 6 MAJOR, +Y, +IC  
 172 MAJOR DOWNWARD, U  
 170 MAJOR UPWARD, U  
 172 MAJOR DOWNWARD, U  
 20064 END 64

V  
 4 MAJOR, +Y, -IC  
 172 MAJOR DOWNWARD, U  
 6 MAJOR, +Y, +IC  
 40027 MINOR STROKE (OPTIONAL)  
 20045 END 45

W  
 6 MAJOR, +Y, +IC  
 32 MAJOR DOWNWARD  
 6 MAJOR, +Y, +IC  
 72 MAJOR DOWNWARD, U  
 6 MAJOR, +Y, +IC  
 40027 MINOR STROKE (OPTIONAL)  
 20046 END 46

X  
 32 MAJOR DOWNWARD  
 0 PEN LIFT  
 1632 MAJOR DOWNWARD X  
 20067 END 67

Y  
 40027 MINOR STROKE (OPTIONAL)  
 33 MINOR DOWNWARD  
 164 MINOR, +Y, +IC, U  
 32 MAJOR DOWNWARD  
 1670 MAJOR DOWNWARD X  
 20070 END 70

Z  
 16 MINOR, +X  
 32 MAJOR DOWNWARD  
 16 MINOR, +X  
 20051 END 51

Z  
 40027 MINOR STROKE (OPTIONAL)  
 33 MINOR DOWNWARD  
 40022 MINOR, +Y, -IC (OPTIONAL)  
 02 MAJOR DOWNWARD  
 1634 UPSTROKE X  
 20051 END 51

I  
 31 MINOR UPWARD  
 1 MAJOR, +Y, -IC  
 0 PEN LIFT  
 36 RIGHT STROKE  
 20075 END 75

2

22 MINOR, +Y, +1C  
32 MAJOR DOWNWARD  
36 RIGHT STROKE  
20002 END 2

3

40036 RIGHT STROKE (OPTIONAL)  
23 MINOR, -Y, +1C  
40031 MINOR UPWARD (OPTIONAL)  
20023 END 23

4

1 MAJOR, -Y, 0C  
0 PEN LIFT  
17 MINOR, -Y, 0C  
36 RIGHT STROKE  
20004 END 4

4

17 MINOR, -Y, 0C  
36 RIGHT STROKE  
0 PEN LIFT  
1 MAJOR, -Y, 0C  
20004 END 4

5

17 MINOR, -Y, 0C  
23 MINOR, -Y, +1C  
0 PEN LIFT  
36 RIGHT STROKE  
20025 END 25

6

3 MAJOR, -Y, +1C  
24 MINOR, +Y, +1C  
21 MINOR, -Y, -1C  
20026 END 26

7

36 RIGHT STROKE  
32 MAJOR DOWNWARD  
20007 END 7

8

7 MAJOR, -Y, -2C  
1730 MAJOR UPWARD, C, X  
20010 END 10

9

21 MINOR, -Y, -1C  
124 MINOR, +Y, +1C, C  
32 MAJOR DOWNWARD  
20031 END 31



0  
 40037 LEFT STROKE (OPTIONAL)  
 33 MINOR DOWNWARD  
 74 UPSTROKE, R  
 0 PEN LIFT  
 32 MAJOR DOWNWARD  
 20040 END 40  
 (space)  
 -  
 36 RIGHT STROKE  
 20020 END 20  
 , (insert function)  
 34 UPSTROKE  
 35 DOWNSTROKE  
 20037 END 37  
 v (transpose function)  
 34 UPSTROKE  
 40036 RIGHT STROKE (OPTIONAL)  
 35 DOWNSTROKE  
 34 UPSTROKE  
 20045 END 45  
 \ (delete function)  
 37 LEFT STROKE  
 20052 END 52  
 . (period)  
 25 MICRO  
 0 PEN LIFT  
 35 DOWNSTROKE  
 20053 END 53  
 < (shift up)  
 2 MAJOR, +Y, 0C.  
 20074 END 74  
 > (shift down)  
 1 MAJO , 0C  
 20072 END  
 \* (Middle dot)  
 25 MICRO  
 0 PEN LIFT  
 36 RIGHT STROKE  
 20054 END 54

**(comma)**

23 MINOR, -Y, +1C  
0 PEN LIFT  
35 DOWNSTROKE  
20073 END 73

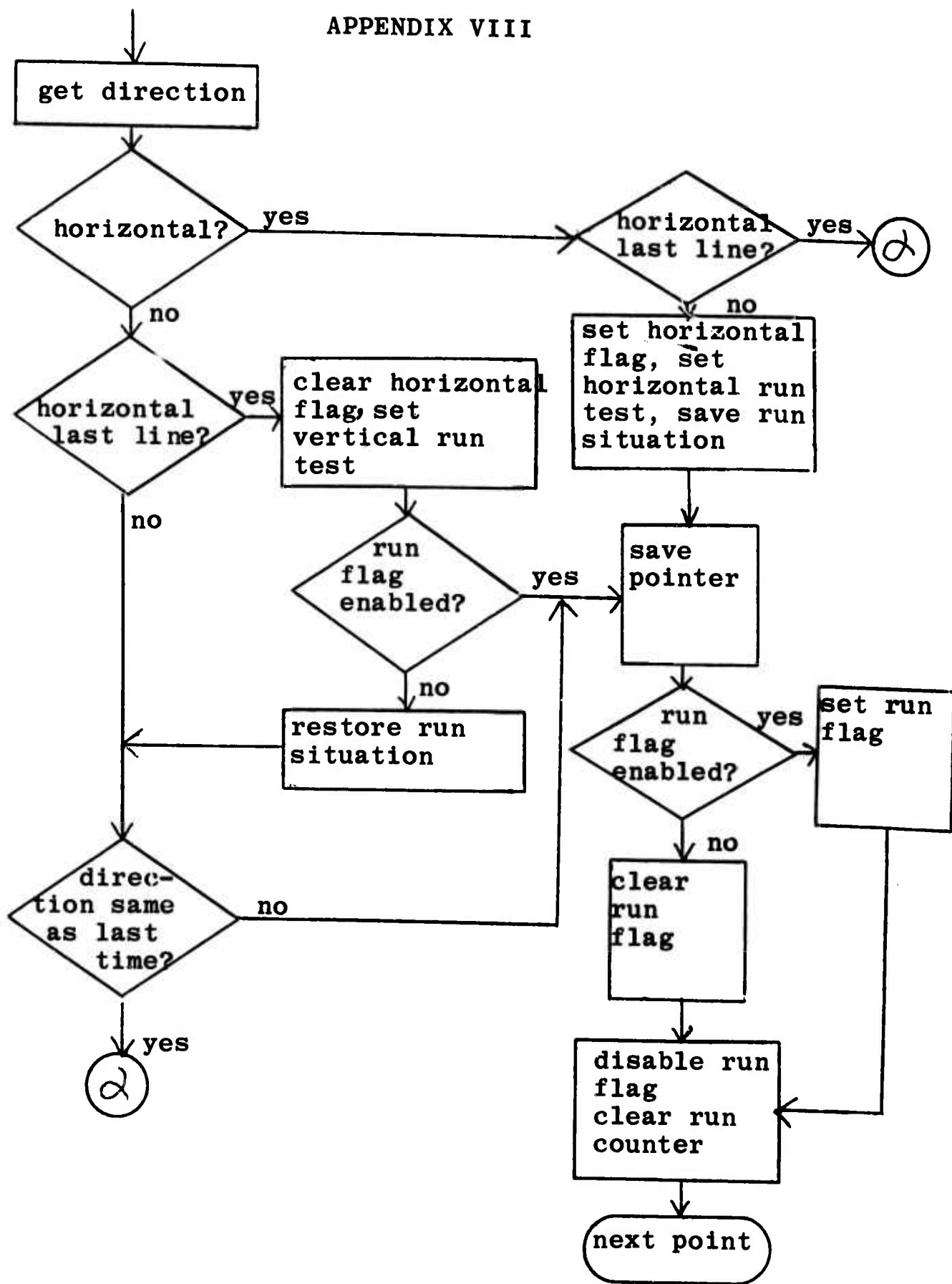
**(backspace)**

36 RIGHT STROKE  
0 PEN LIFT  
35 DOWNSTROKE  
20076 END 76

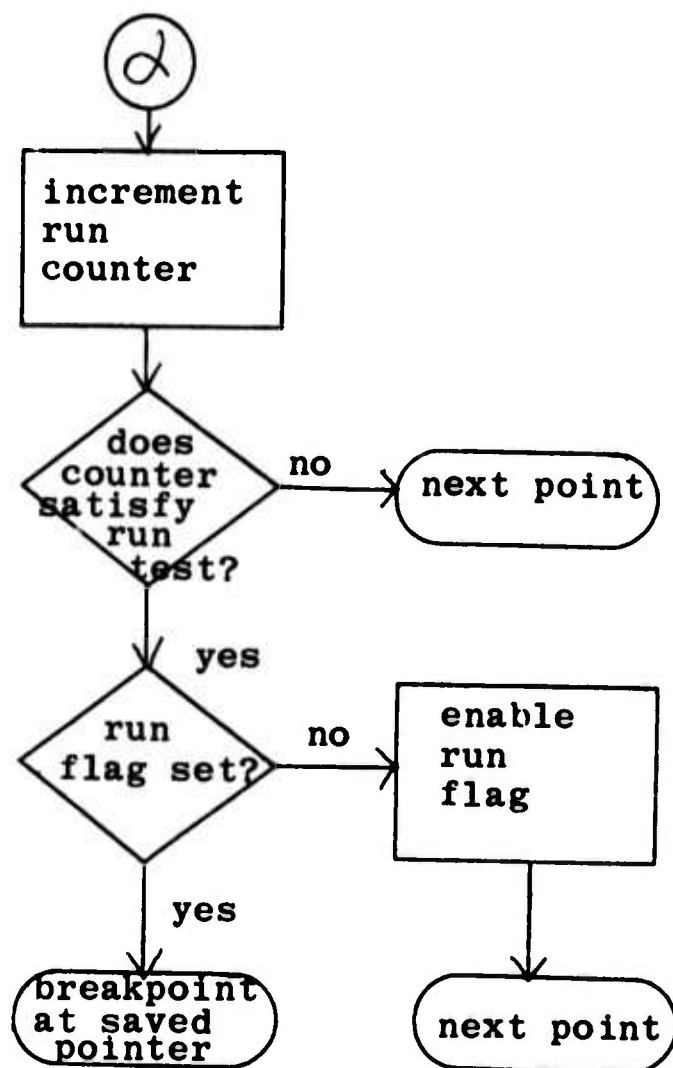
**(carriage return)**

5 MAJOR, -Y, +1C  
20077 END 77

# APPENDIX VIII



FLOW CHART OF HORIZONTAL SEGMENTING ALGORITHM



#### Enabling and setting of run flag:

The run flag is enabled (allowed to be set) when a long enough run occurs. If it is already enabled, it is set, indicating two successive long enough runs in different directions, and thus a breakpoint.

next point

indicates the next point should be accessed and the program should return to the beginning of the algorithm.

FLOW CHART OF HORIZONTAL SEGMENTING ALGORITHM (Continued)

## BIBLIOGRAPHY

- M. I. Bernstein, An On-Line System for Utilizing Hand-Printed Input, SDC Tech. Memo TM-3052/000/00, July 11, 1966.
- M. Eden, "On the Formalization of Handwriting," Proceedings of Symposium in Applied Mathematics, Vol. XII, American Mathematical Society, Providence, 1961, pp. 83-88.
- M. Eden, "Handwriting and Pattern Recognition," IRE Trans. on Information Theory, IT-8, Feb. 1962, pp. 160-166.
- M. Eden and M. Halbe, "The Characterization of Cursive Writing," Information Theory, Fourth London Symposium, C. Cherry, ed., Butterworths, London, 1961, pp. 287-299.
- F. N. Freeman, Guiding Growth in Handwriting: Reference Manual for Teachers, Grades Five through Eight, Zaner-Bloser, Columbus, 1959.
- H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," IRE Trans. on Electronic Computers, EC-10, June 1961, pp. 260-268.
- L. S. Frishkopf and L. D. Harmond, "Machine Reading of Cursive Script," Information Theory, Fourth London Symposium, C. Cherry, ed., Butterworths, London, 1961, pp. 300-316.
- G. F. Groner, "Real Time Recognition of Hand-Printed Text," AFIPS Conference Proceedings, Vol. 29, Spartan Books, Washington, 1966, pp. 591-601.
- F. Kuhl, "Classification and Recognition of Hand-Printed Characters," IEEE International Conference Record, Vol. II, Part 4, 1963, pp. 75-93.
- N. Lindgren, "Machine Recognition of Human Language - Part III - Cursive Script Recognition," IEEE Spectrum, May 1965, pp. 104-116.
- P. Mermelstein and M. Eden, "Experiments on Computer Recognition of Connected Handwritten Words," Information and Control, V7, 1964, pp. 255-270.
- P. Mermelstein and M. Eden, "A System for Automatic Recognition of Handwritten Words," AFIPS Conference Proceedings, (1964 FJCC), Vol. 26, Part 1, Spartan Books, Baltimore, 1964, pp. 333-342.
- W. R. Nugent, "Handwritten Non-Phonetic Alphabets for Real Time Man-Computer Communication," Digest, IEEE 1967 International Conference on Communication, Minneapolis, June 1967.
- A. N. Palmer, The Palmer Method of Business Writing, Palmer, Chicago, 1949.

W. Teitelman, "Real Time Recognition of Hand Drawn Characters,"  
AFIPS Conference Proceedings, (1964 FJCC), Vol. 26, Part 1,  
Spartan Books, Baltimore, 1964, pp. 559-576.

Zaner-Bloser, Penmanship Charts, Columbus.

**BLANK PAGE**



UNCLASSIFIED

Security Classification

## DOCUMENT CONTROL DATA - R &amp; D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

## 1. ORIGINATING ACTIVITY (Corporate author)

Inforonics, Incorporated  
806 Massachusetts Avenue  
Cambridge, Massachusetts 02139

## 2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

## 2b. GROUP

N/A

## 3. REPORT TITLE

MOSAIC - THE IMPROVED EDITING OF SCIENTIFIC TEXT BY HAND-DRAWN COMMANDS  
AND DATA: A TECHNIQUE FOR RAND TABLET AND CRT DISPLAY

## 4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

None

## 5. AUTHOR(S) (First name, middle initial, last name)

Seymour R. Friedman  
Douglas A. Campbell  
Leonard G. Fehskens

## 6. REPORT DATE

14 October 1968

## 7a. TOTAL NO. OF PAGES

73

## 7b. NO. OF REFS

one

## 8a. CONTRACT OR GRANT NO.

AF 19(628)-5992

## 8b. PROJECT NO.

## 9a. ORIGINATOR'S REPORT NUMBER(S)

ESD-TR-68-422

## 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

## 10. DISTRIBUTION STATEMENT

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of Hq ARPA (TIO).

## 11. SUPPLEMENTARY NOTES

## 12. SPONSORING MILITARY ACTIVITY

Command Systems Division, Electronic  
Systems Division, AFSC, USAF,  
L G Hanscom Field, Bedford, Mass. 01730

## 13. ABSTRACT

A method for improved editing of scientific text by hand-drawn commands and data is considered. The most distinguishing feature of the method is that a single quick editing mark over a data segment simultaneously identifies both operator and operand. The method investigated employs a RAND Table for the entry of hand-drawn cursive symbols to the computer, an input-only keyboard for large strings of text and a storage-type CRT display unit for displaying the text being edited. The recognition of cursive writing is performed by using geometric-topological invariants of stroke succession. In addition, a design for an editing console is also discussed. In this design, an attempt is made to arrange the different system components to provide for efficient editing. The conclusions derived from the use of the system during its development are presented, along with several areas for further investigation.

14.	KEY WORDS	LINK A		LINK B		LINK C	
		ROLE	WT	ROLE	WT	ROLE	WT
	Digital Computers						
	Text Editing						
	Character Recognition						
	Graphics						
	Cathode Ray Tube						
	RAND Tablet						
	Editing Consoles						